Quantitative Modeling of Software Reviews in an Industrial Setting

Oliver Laitenberger
Fraunhofer Institute for
Experimental Software Engineering
Sauerwiesen 6
D-67661 Kaiserslautern
Germany
Oliver.Laitenberger@iese.fhg.de

Marek Leszak, Dieter Stoll
Lucent Technologies Network
Systems GmbH
Thurn-und-Taxis-Strasse 10
D-90411 Nuernberg
Germany
{mleszak, dieterstoll}@lucent.com

Khaled El Emam
National Research Council,
Institute for Information
Technology, Building M-50,
Montreal Road, Ottawa Ontario
Canada K1A OR6
Khaled.El-Emam@iit.nrc.ca

Abstract

Technical reviews are a cost-effective method commonly used to detect software defects early. To exploit their full potential, it is necessary to collect measurement data to constantly monitor and improve the implemented review procedure. This paper postulates a model of the factors that affect the number of defects detected during a technical review, and tests the model empirically using data from a large software development organization. The data set comes from more than 300 specification, design, and code reviews that were performed at Lucent's Product Realization Center for Optical Networking (PRC-ON) in Nuernberg, Germany. Since development projects within PRC-ON usually spend between 12% and 18% of the total development effort on reviews, it is essential to understand the relationships among the factors that determine review success. One major finding of this study is that the number of detected defects is primarily determined by the preparation effort of reviewers rather than the size of the reviewed artifact. In addition, the size of the reviewed artifact has only limited influence on review effort. Furthermore, we identified consistent ceiling effects in the relationship between size and effort with the number of defects detected. These results suggest that managers at PRC-ON must consider adequate preparation effort in their review planning to ensure high quality artifacts as well as a mature review process.

Keywords: Technical reviews, success factors, path analysis.

1. Introduction

Technical reviews[1] are a proven approach that enables the detection and correction of defects in software artifacts as soon as these artifacts are created. They not only improve the quality of the artifacts but also help software development organizations reduce their cost of producing software. This stems from the fact that reviews allow the identification of defects at a stage where they are easier and relatively inexpensive to correct, thereby causing the development process to avoid additional rework penalties associated with defect detection at later test and integration stages.

At Lucent's Product Realization Center for Optical Networking (PRC-ON) in Nuernberg, Germany, the review process is an essential element of the Standard Development Process (SDP). The review process has been defined based on worldwide published review processes [8] and the same review process has been applied to all major large-scale development projects since 1995.

Today, reviews at PRC-ON usually consume between 12% and 18% of the total development effort. These costs include quality assurance (i.e., milestone) reviews as well as technical reviews on documents, software sources, and other artifacts of the development process. This makes the case to use the collected review measurement data to understand and establish relationships among the factors that determine review success. A good understanding builds the foundation for optimizations and improvements of the current review approach as well as better management of it.

In this paper we postulate a theoretical model that specifies factors that have an impact on the extent of review success at PRC-ON, and the functional form of these relationships. Review success is defined as the number of defects detected during the review. The factors that we focus on are those that have been suggested in the literature to have a strong influence on review success: preparation effort and the size of the inspected document.

Our analysis is based on data from more than 300 specification, design, and code reviews performed at PRC-ON. This affords us the opportunity to identify consistent relationships across different types of reviews, unlike previous research which focused largely on code reviews only, e.g., [4] [15]. Failure of the model to fit the collected data results in its falsification in this environment, while a good fit allows the model to survive, but not be proven, since other models might

1. Other terms such as formal technical review or software inspection could have been used here.

provide equal or better fits.

Briefly, our results indicate that artifact size has only limited influence on both preparation effort of reviewers and the number of detected defects. It is rather the effort spent for preparation that is found to be a more significant factor influencing the number of detected defects. Furthermore, we identified consistent ceiling effects in the relationship between size and effort with the number of defects detected. The consistency of these results across different artifact types has not been reported previously.

The paper is organized as follows. Section 2 elaborates in more detail the theoretical model we distilled from the literature. Section 3 presents the research method. The presentation includes a discussion of the Lucent environment, the review process at Lucent, and the measurement procedure, and the analysis techniques. Section 4 presents the results. Section 5 briefly discusses the findings. Finally, Section 6 concludes with a summary and directions for future work.

2. Background

2.1 Motivation

As a discipline matures, empirical work should be performed to investigate the veracity of postulated theoretical models. A theoretical model provides one or more hypothetical predictions that may be tested by collected data. Testing a theoretical model is particularly beneficial for the investigation of software reviews for the following three reasons. First, practitioners as well as researchers gain insight into the main factors influencing review success. This insight can and should be the foundation as well as the trigger for improvement activities. Second, the models offer researchers the unique opportunity to integrate their own work into a broader context and to highlight his or her methodological or empirical contribution in a systematic manner. Finally, in the long run, the constant refinement of those models allows a systematic accumulation of knowledge, which makes technical reviews an even more effective approach for overcoming software quality deficiencies and cost overruns.

For the development of theoretical models for review success, a researcher must keep in mind that the most successful review approach is the one that helps to find most of the defects in the reviewed artifact and has an optimal cost/benefit ratio[1]. For both purposes, the number of detected defects is one of the key measures. But it needs to be interpreted in a meaningful way because a high number of defects may either indicate highly effective reviews or high defect injections. A low number of defects, on the other hand, may indicate poor reviews or a high maturity of the software development organization. If further variables are measured and the variables are looked at together within the context of a theoretical model, we can extract useful information for review improvement.

Although there might be a limitless number of factors that induce variations on the number of defects, the body of existing review work has in general revealed two major ones: Preparation effort and size of the reviewed artifact. However, the many empirical findings, such as the ones consolidated in [12], have rarely been investigated in the context of a theoretical model. An exception is Porter et al.'s work [15]. They present a cause-and-effect diagram of the review process as a starting point for understanding the sources of variations in code reviews within one particular development project. They focused their attention on three factors, that is, reviewers, authors, and code units, which they found to be significant. Moreover, they stated that "further investigation is needed to quantify the effect of preparation time on defects found". This will be examined as part of our study. Hence, we elaborated on elements of Porter et. al's model and present some further results.

2.2 A Theoretical Model for Explaining the Number of Detected Defects

The number of detected defects can be modeled in the form of a path diagram [14] as presented in Figure 1.

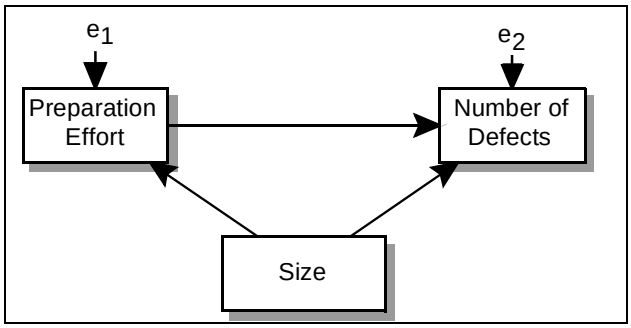

Figure 1: Path Diagram For Explaining Defects

An arrow linking a given pair of variables (X,Y) indicates the assumption about a direct causal link between these variables. Hence, it must be interpreted as a hypothetical statement of the form "an increase in X is expected to produce (cause) an increase in Y". The terms e_1 and e_2 represent error terms of the model.

The hypothetical model in Figure 1 can be explained in the following manner:

1. A third criteria according to Votta is the duration of a review [23]. However, this criteria is not considered in this study.

Hypothesis H_1: The larger the effort for defect detection (i.e., the preparation effort) the more defects are detected.

We assume here that preparation effort is an important factor on the number of defects detected. Preparation effort is the effort the reviewers individually spend for scrutinizing the artifacts for defects. The relationship between review effort and number of defects is due to the effect of more reviewers inspecting a document (the more reviewers the more defects are found), and also due to the increased effort spent by the reviewers looking for defects. We examine each of these in turn.

We make the assumption that all reviewers have the same probability, p, of detecting a defect. The probability p can be interpreted as follows: if we sample defects with replacement from the set of defects in the document then in the long run the proportion of times where a defect that is detected is selected is equal to p. This interpretation has the convenient property that it is congruent with the notion of inspector effectiveness. This assumption is not necessary and our inferences will still hold otherwise, however it does lend itself to algebraic simplicity.

The probability for a review team with k reviewers of finding a defect is given by:

$$\sum_{i=1}^{k} \binom{k}{i} p^i (1-p)^{k-i} \qquad \text{(eq. 1)}$$

For example, for $k = 2$ the probability for a reviewer of finding a defect is p and of not finding a defect is $1-p$. The probability of either reviewer finding a defect or both finding a defect is given by

$$p(1\text{-}p) + (1\text{-}p)p + p^2 \qquad \text{(eq. 2)}$$

which is the equation that we have above. Let us say that $p = 0.4$, then the plot of the probability of the review team detecting a defect against the number of reviewers is shown in Figure 2.

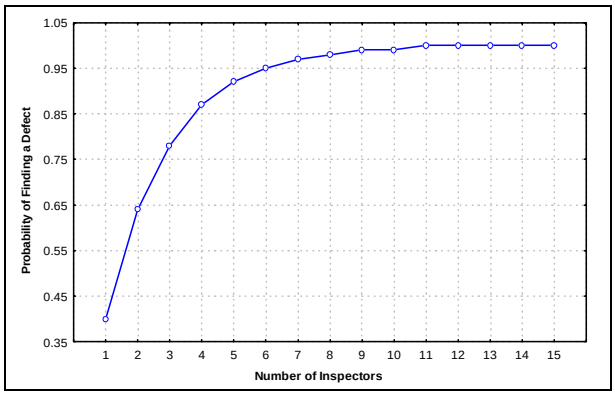

Figure 2: Defect Detection Probability against the Number of Reviewers

As can be seen from the plot, the probability of defect detection would increase as the number of reviewers increases, but there is a ceiling effect. There comes a point were the addition of more reviewers will lead to relatively less defects being detected.

The stated relationship for individual reviewer effort can be explained by the fact that more effort allows for a better understanding of the reviewed artifact, which directly translates to more detected defects. However, we assume a ceiling effect, i.e, the number of detected defects levels off after a certain amount of effort has been spent for preparation. The ceiling effect may be explained by the following three reasons. First, after a certain amount of effort expended, we would expect the reviewers to get tired, and even if they spend further effort, they will not be able to find many more defects. Second, other project goals, such as important deadlines and the resulting time pressure, are perceived more important and may set a natural threshold for the preparation effort of reviewers. And third, defects detected later in the preparation process are usually more difficult to detect. Hence, more effort must be invested for their detection.

The validity of the hypothetical individual effort relationship has been observed several times in practice. Porter et. al [18] investigated code reviews and found that individual preparation has a significant impact on the number of detected defects. This led them to conclude that better preparation techniques (i.e., reading techniques [1]) rather than review process variations may significantly improve current review implementations [17]. Christenson et. al state that effective code reviews were those that met the recommended preparation effort [4]. Raz and Young corroborated this finding for design and code reviews [18]. They found reviews without sufficient preparation to be not fully effective.

The hypothesis, however, makes the assumption that defect detection is more an individual than a group activity. We are aware that the organization of the defect detection activity of reviews is still debated in the literature. More specifically, the issue is whether defect detection is more an individual activity and hence should be performed individually, or whether defect detection is a group activity and should therefore be conducted as part of a group meeting. Following Fagan's approach [7] a group meeting provides a synergy effect. This leads to the assumption that group meetings help detect more defects. In this case one must rather consider the preparation effort and the meeting effort. A causal model to explain the latter can be found in [21]. However, others found little synergy in a meeting-based organization [10] [23]. Instead they point out that individual preparation is key. Following this argumentation, a group meeting does not necessarily help detect more defects. Since we observed the latter at PRC-ON, we focus on preparation

effort in this study.

Hypothesis H_2: The larger the size of a reviewed artifact, the more defects are detected in its review.

We assume here that the size of the reviewed artifact is a second crucial factor for the number of defects found. The rationale behind this relationship is that larger artifacts are expected to be more complex and due to their size, create more opportunities for defects to be introduced. Therefore a relationship between size and the number of defects is anticipated. However, a ceiling effect can also be observed in this relationship. Empirical evidence implies that larger artifacts are proportionally less defect-prone than smaller ones. Les Hatton describes several empirical studies in which this effect could be observed [9]. Moreover, Kelly et al. state that increasing the number of pages at one time decreases the number of defects found [11]. Christenson et. al demonstrated the variance of the defect density to be inversely proportional to the size of the unit of code [4][1]. Yet, most of these findings are limited to code artifacts and its an open question whether they scale up to artifacts produced early in the life cycle, such as specification or design documents.

Hypothesis H_3: The larger the size of a reviewed artifact, the more preparation effort is spend.

We assume here that the preparation effort is primarily determined by the size of the reviewed artifacts. A relationship between size and effort is expected for two reasons. First, whenever a larger document is being reviewed there is a tendency to assign more reviewers to work on it, and hence the effort will increase. But also, at the individual level, we would expect that a larger document would take longer to review since the information content is larger.

Again we would expect a ceiling effect for which fatigue effects and the loss of motivation are possible explanations. Because of the ceiling effect larger artifacts may tend to receive proportionally less preparation effort than small ones (with its detrimental effect on the number of detected defects). The latter has been reported, for example, by Christenson et al. [4].

3. Research Method

This section describes the research context at PRC-ON as well as the measurement, data collection, and data validation approach. Finally, it presents our analysis method.

1. However, Christenson et. al's findings need to be interpreted carefully since they used the size variable on both sides of the equation.

3.1 Development Environment

The basic workflow around the review process at PRC-ON is depicted in Figure 3. The standard development process (SDP) requires certain reviews in certain stages of a development project. All artifacts and their quality checking activities (reviews, testing) must be planned and scheduled by the respective teamleader in charge of developing a subsystem. Once a deliverable is ready (from a developer' s point of view), the teamleader delegates review control to a qualified moderator. The moderator is responsible for the selection of the right mix of experts for a review team, and for the success and performance of a review.

There are no guidelines at PRC-ON on how many reviewers to invite to a review. Hence, the number of reviewers is determined by the availability of people, project constraints, and the reviewed artifact itself. The reviewers themselves usually have a high level of experience and can be considered experts for the reviewed artifact. They do not use any particular reading technique as suggested in [1].

Apart from the moderator role, other roles in the review process are

- recorder - records defects in the defect list.
- checker - verifies correctness and completeness of the reworked artifact after the review meeting,
- reviewer - probes the artifact for defects and reports them in the review meeting. The author or the moderator may also act as a reviewer.

The review process itself is defined in a number of standard phases, i.e., planning, kick-off and overview, preparation (of reviewers), defect logging in a group meeting, rework, and checking.

The review process is mandatory for all newly developed and significantly changed artifacts. However, since all changes must be guarded by modification requests (MRs), not every small change triggers a review.

3.2 Measurement

The metrics that are collected in software component reviews at PRC-ON are review type, effort, size, defects, and the number of reviewers. These can be characterized as follows:

- Type of review: Component specification reviews, component design reviews, or code reviews.
- Size of reviewed artifact - measured in document pages or noncommentary source lines (NCSL).

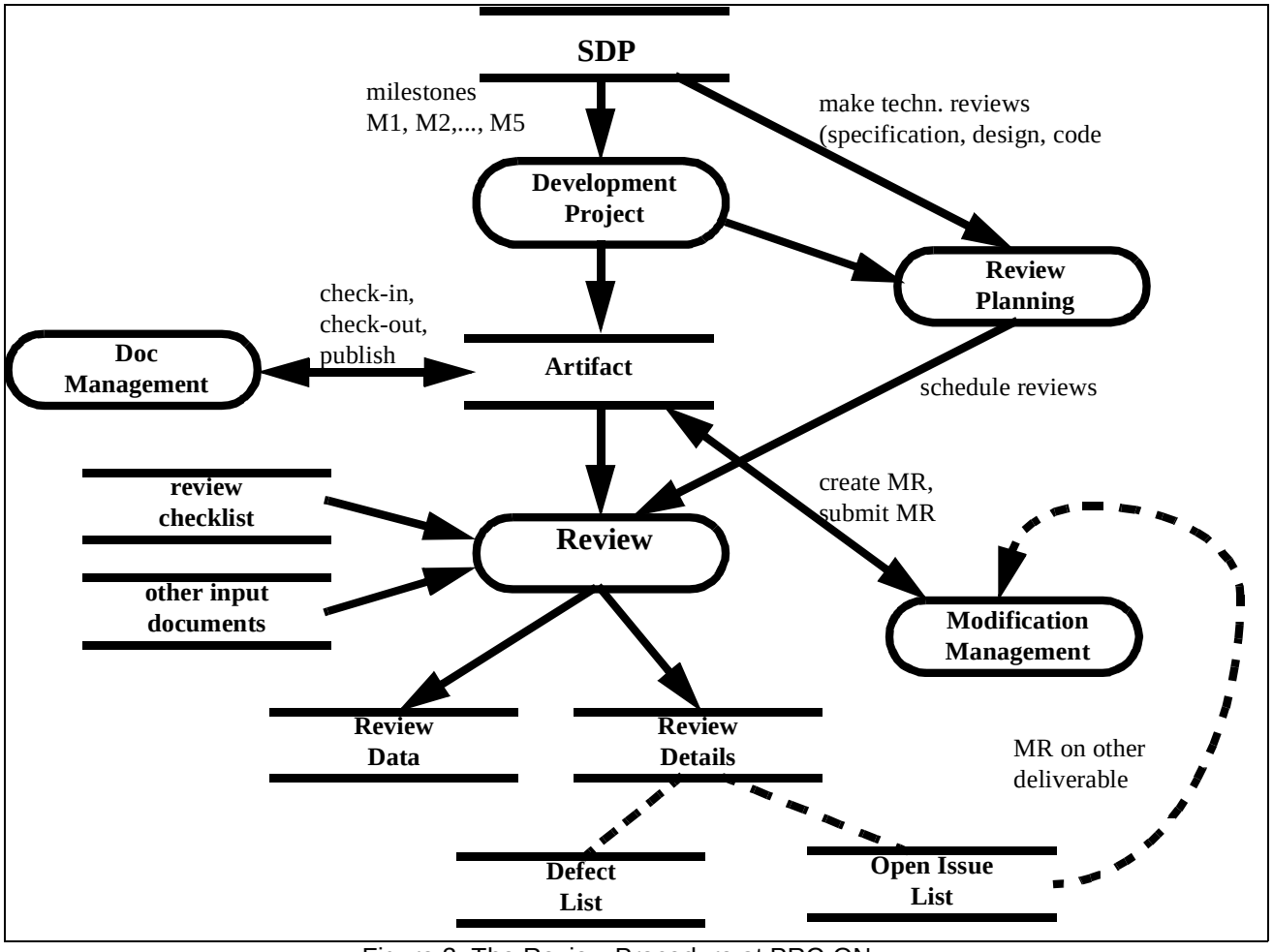

Figure 3: The Review Procedure at PRC-ON

- Preparation effort of all reviewers
- Meeting effort (for the kick-off and the review meeting)
- Number of reported defects. Although a distinction is made between major and minor defects, we focus in this study on the total number of defects. This stems from the fact that we want to determine the influential factors on review success rather than perform a cost/benefit analysis for reviews in which a distinction according to defect criticality is helpful. Moreover, minor defects were found to also provide a contribution towards higher quality [22].
- Number of reviewers for a review team

Data on these attributes are the information available that can be used to validate the theoretical model stated in Section 2.

3.3 Data Collection Procedure

For large-scale system development, the implemented review procedure can only be monitored and controlled if adequate tool support is available. Hence, tool support for the administrative part of technical reviews is provided. The tool is called QADM [12]. QADM is a GUI-based tool that

- supports review planning, recording and tracking.
- maintains all data collected during reviews, by keeping a review database.
- allows for the collection of review measures and the generation as well as the visualization of initial analysis results. Hence, it stores all information produced during a review - except the so-called defect list. Only the number of defects is stored in QADM, not the detailed description of each defect. The defect list is fed into the electronic review minutes later or is archived as a manual record in a central repository.

- allows for powerful and efficient searches through the review database.
- generates review-based quality records (e.g., for ISO 9001 certification).

3.4 Data Validation

For any kind of analysis effort, the quality of data is essential - otherwise the 'junk-in / junk-out' syndrome prevents researchers and practitioners from getting any meaningful result. At PRC-ON, one member of the review team enters the data in QADM. Since manually collected data are at a higher risk of containing errors, we performed an extensive validation of the entered review data to detect and remove missing, incorrect, and inconsistent entries. In cases where we could not clarify seemingly inaccurate entries, we decided to remove them. After data validation 340 entries remained for analysis (i.e. data from 145 specification reviews, 94 design reviews, and 101 code reviews).

3.5 Path Analysis

The method that we employ to test the described theoretical model is path analysis [14], [19]. This method allows us to depict the whole model on one diagram and to compute direct and indirect effects of the variables.

The path coefficients in the model are derived using ordinary least squares regression analysis [2], [5]. We use logarithmic transformation due to our expectations of ceiling effects. Mathematically, such a relationship between two variables A and B would be expressed as:

$$A = \alpha * B^{\beta} \qquad \text{(eq. 3)}$$

where $0 < \beta < 1$. To examine this model with ordinary least squares regression, logarithmic transformations can be performed on each side of the regression equation to obtain the following linear model:

$$\ln(A)=\ln(\alpha)+\beta*\ln(B) \qquad \text{(eq. 4)}$$

This is the approach of choice for our analysis. The path coefficients we consider for our theoretical model are the beta coefficients of the regression analysis. Beta coefficients indicate the average standard deviation change in the dependent variable with a standard deviation change in one independent variable, when others are held constant. The use of standardized coefficients allows the comparison of the magnitude of relationships. Following standard procedures, we used statistical significance testing with a significance threshold (alpha level) of 0.01 to determine whether a variable is statistically significant.

The path model that we test is depicted in Figure 1. The *e* values are the residual terms. The larger these values are the more variation on that variable that is not explained by the model. The errors are calculated as $\sqrt{1 - R^2}$.

The fact that we test the model for reviews in three phases of the life cycle allows us to check for consistency in the model. If the results are consistent then we would be able to generalize further our understanding of software reviews (note that in previous work researchers commonly focus on reviews within a single phase).

4. Results

We start the analysis effort with an exploratory analysis using descriptive techniques to gain some initial insights into the data. We look at the various review types and compared them with already existing findings from the review literature. Then we present the results of the path analysis.

4.1 Descriptive Statistics

4.1.1 Number of detected defects

Figure 4 depicts the number of defects that are detected in the different types of reviews. The box represents the interquartile range (i.e., 50% of all observations fall within this range), while the whiskers represent the minimum and maximum value.

As Figure 4 shows, defect distribution is consistent among review types. The median value of a specification, design, and code review is 12, 15, 14, respectively. While the interquartile range of specification reviews and design reviews is 21 defects, it is slightly lower for code reviews (17 defects).

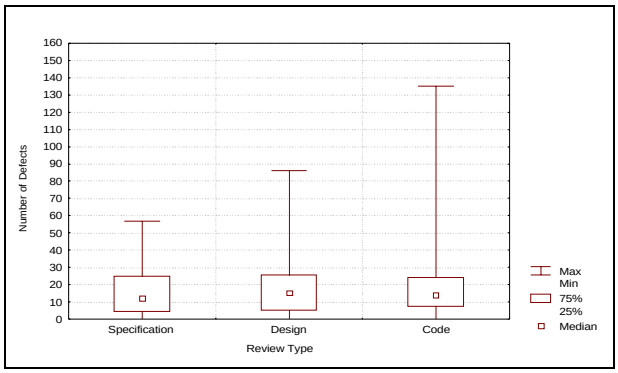

Figure 4: Number of Detected Defects

The presented findings establish an organization-specific baseline for PRC-ON against which to compare any improvement that promises to increase the number of defects detected. However, an evaluation in the context of other

review work is difficult because most studies only focus on code reviews and often do not present the number of defects found, but rather some summary statistics [20] [24].

4.1.2 Review Effort

Figure 5 shows the preparation effort distribution as well as the distribution of the total effort spent on reviewing the various artifacts. The total effort includes the preparation as well as the meeting effort of all review participants.

Figure 5 reveals that reviewers involved in any kind of review usually spend between 2 and 8 hours for preparation (independent of the number of reviewers) and between 4 and 14 hours for the total review. The effort distribution looks similar for the different types of reviews.

The results show that the review of artifacts in early phases (i.e., specifications) does not significantly consume more effort than code artifacts. The median effort for specification, design, and code reviews (7, 6, 8 person hours) as well as the upper quartile ranges (11, 10, 13 person hours) provide a lower threshold for managers on how much effort the review of a particular artifact type may at least consume in future projects.

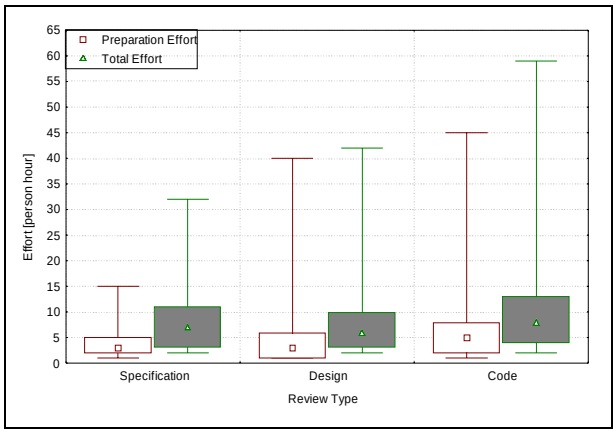

Figure 5: Review Effort

In this study we focus on preparation effort as an important parameter to optimize. Figure 6 depicts the relationship of preparation versus meeting effort. It shows that most of the reviews consume at least as much effort in preparation than in the meeting. Surprisingly, the preparation/meeting-ratio is highest for code reviews. This may be explained by the following two reasons. First, the reviewers do not spend as much effort for the preparation of design or specification reviews. And second, the meetings for specification and design are more effort consuming since the discussion of defects detected takes longer.

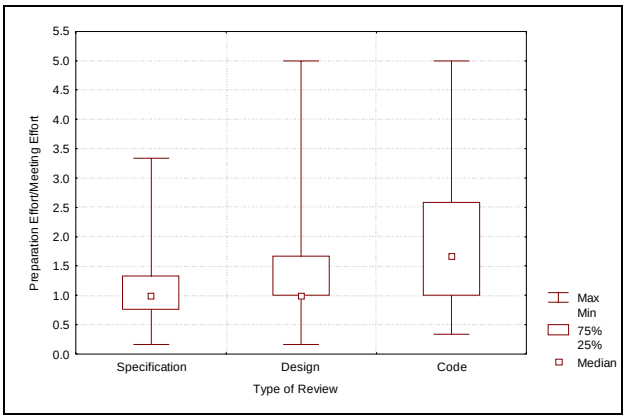

Figure 6: Relationship of Preparation Effort and Meeting Effort

4.1.3 Size

The unit of size for specification and design artifacts is pages whereas for code artifacts, it is noncommentary source lines of code. Since the measurement units are different for specification/design documents and code components, we present two graphs. Figure 7 exhibits the size distribution across the reviewed artifact types.

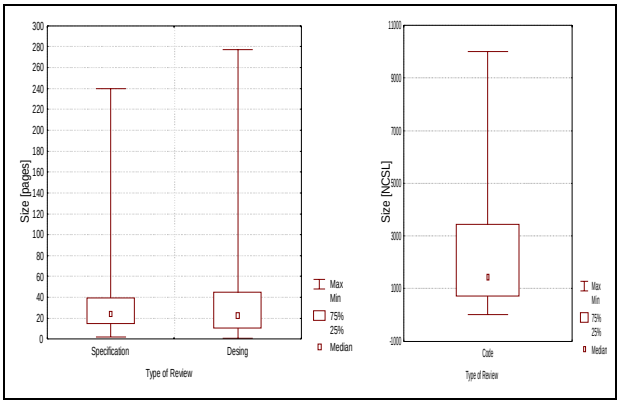

Figure 7: Size of Reviewed Artifacts

Figure 7 reveals that the median size of a specification is 24 pages, whereas it is 22.5 pages for design artifacts. Most of the reviewed documents are smaller than 50 pages. The median size for code components is 1450 NCSL and most of the reviewed code components are lower than 3420 NCSL. Documents of this size are neither too large nor too small for review and are within the range of the ones reported in other studies [8].

4.1.4 Number of Reviewers

In addition to defect, size, and effort distribution, we also

investigated the number of reviewers. Figure 8 shows how many reviews have been performed for each artifact type with a specific number of reviewers.

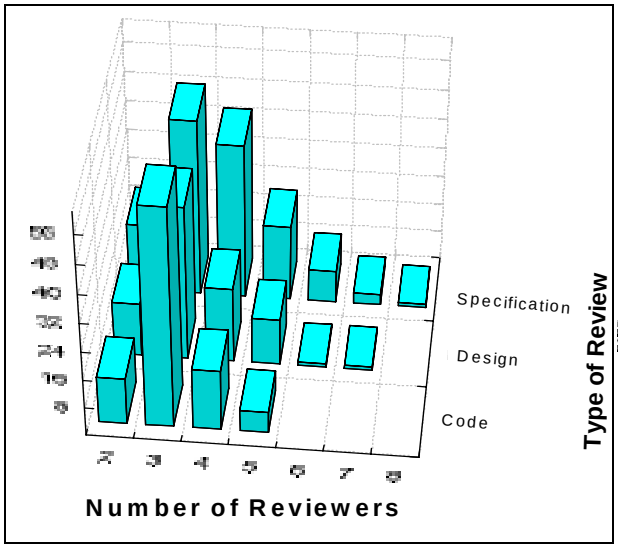

Figure 8: Histogram of the Number of Reviewers for the Different Types of Reviews

According to Figure 8, most of the reviews were performed with 3 reviewers. Specification reviews often involve a higher number of reviewers. This emphasizes the importance of the specification phase for design and coding.

The optimal number of reviewers is still debated in the literature. This debate boils down to the question whether involving more reviewers helps detect more defects. Surprisingly, there are few consistent results so far. Weller presents some data from a field study using three to four reviewers [24]. Madachy presents data showing that the optimal size is between three and five people [13]. Bourgeois corroborates these results in a different study [3]. Porter et al.'s recent experimental results, however, suggest that reducing the number of reviewers from 4 to 2 may significantly reduce effort without increasing review interval or reducing effectiveness [16].

4.1.5 Defect Density

Since we assumed that the number of defects is related to the size of the document, we calculated the defect density defined as defects per unit of size. Figure 9 shows the result of this calculation.

Reviews exhibit on average 0.53 defects/page when looking at specification reviews. Design reviews exhibits 0.58 defects/page. Finally, code reviews find 7.0 defects/ KNCSL. The variation seems to be small.

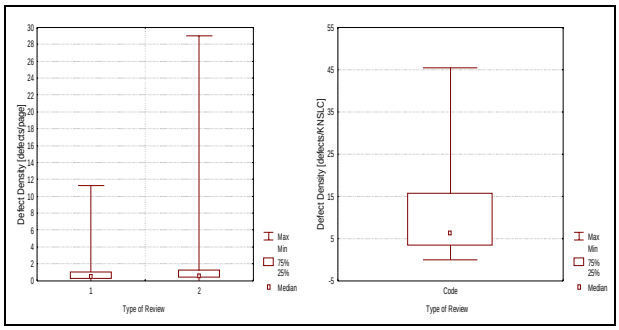

Figure 9: Defect Density

For specification and design artifacts, we did not find comparable figures in the literature. For code artifacts, however, the defect densities are within the reported range of other telecom organizations. Ebert et. al. describe some results from Alcatel Telecom [6]. There, code components exhibit an average defect density of 9 defects/KNCSL. This result supports the initial statement that the review process for code components at PRC-ON belongs to the state of the practice that can be found in the software industry.

4.2 Path Analysis Results

We used regression analysis to investigate the hypothesized model presented in Section 2 and report the beta coefficient. A star indicates whether a beta coefficient is statistically significant.

4.2.1 Specification Reviews

Figure 10 depicts the model for specification reviews. The total effect of size on defects (direct and indirect) is 0.24 (=0.17*0.77+0.11). This value is lower than the direct effect of preparation effort on defects. This shows that preparation effort has a larger impact on the number of defects than size.

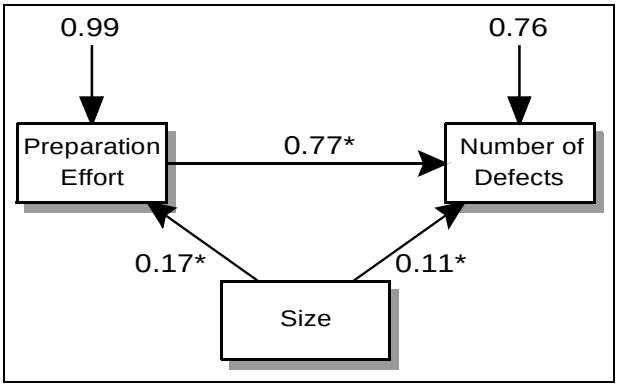

Figure 10: Path Diagram for Specification Reviews

4.2.2 Design Reviews

Figure 11 shows the path model for design reviews. The total effect of size on defects (direct and indirect) is 0.52, which is slightly below the value of preparation effort on the number of defects.

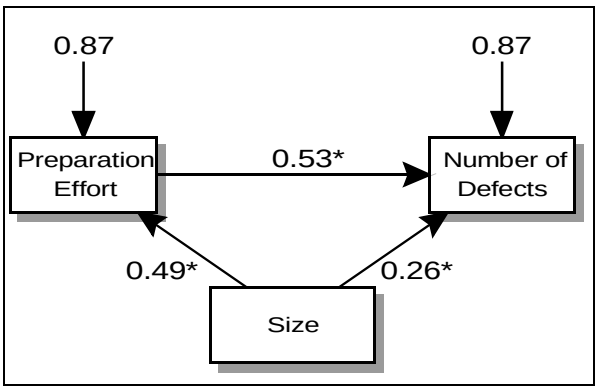

Figure 11: Path Diagram for Design Reviews

4.2.3 Code Reviews

Figure 12 reveals the model for code reviews. The total effect size of size on defects (direct and indirect) is 0.29. Again preparation effort has a larger impact on the number of defects than size. Moreover, size was found not to be a statistically significant factor for the number of defects detected in code reviews.

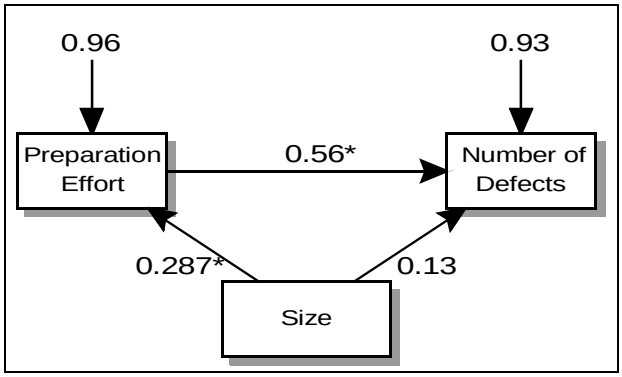

Figure 12: Path Diagram for Code Reviews

4.2.4 Post-hoc Analysis

We also calculated the relationship of preparation effort per reviewer and the number of defects. The results are shown in Table1.

	β	R^2
Specification	0.70*	0.49
Design	0.64*	0.42
Code	0.52*	0.29

Table 1: Preparation Effort per Reviewer versus the Number of Defects found

The results are comparable to the ones we obtain by using the total preparation effort. All path coefficients remain statistically significant. Although this finding shows the importance of preparation we cannot derive whether more reviewers directly pay off in more detected defects or whether the chosen reviewers just need to spend more effort for preparation.

4.2.5 Ceiling Effects

We identified ceiling effects for all the relationships that we investigated across all types of reviews. One can interpret this as stating that additional reviewers and/or individual reviewers expending more effort on preparation reaches a plateau, after which the returns in defect detection diminish. This does not necessarily mean that, for example, many reviewers should not be used (especially if reliability is a high concern), only that this may not be cost effective, and alternative defect detection techniques ought to be investigated.

Furthermore, increasing the size of an artifact results in increased effort up to a certain point, and then the relationship plateaus. This can be an indicator of either that the organization caps the number of reviewers and/or a fatigue effect. Based on our knowledge of the review process, however, the former interpretation is not satisfactory, and therefore, it would seem that there exists a fatigue effect.

5. Discussion

We found evidence supporting the model proposed here and the evidence was consistent for code, design, and specification reviews. Furthermore, we identified a theoretically justifiable functional form for all of the hypothesized relationships.

Our findings suggest that an organization ought to pay attention to preparation effort as a means of controlling the number of defects detected during reviews, and that adding reviewers and increasing preparation effort may not be cost effective since there exists a ceiling effect.

The results here, however, also suggest avenues for

further investigation to better understand the mechanisms that are operating during software reviews. For example, it would be informative to determine whether it is team size or individual preparation effort that is contributing to an increased number of defects detected.

Furthermore, the residual terms remain quite large for all of the three types of reviews. This suggests the existence of more variables that ought to be included to better understand the factors that affect defect detection in software reviews.

A primary candidate that may explain a large amount of variation in the data set is a classification of reviews according to the status of the artifact under review. It is plausible that a review of a newly developed artifact exhibits a different relationships among factors than an artifact that underwent a minor or major modification.

Characteristics of the structure of reviewed artifacts, such as measures of coupling and ambiguity, may be another factor to explain some of the observed variation.

6. Conclusion

Technical reviews are considered one of the most effective methods for software quality improvement. To exploit their full potential, they need to be constantly monitored and optimized. In this paper, we presented a path analysis to better understand review success factors. We found preparation effort is a more influential factor on the number of defects detected than artifact size. Hence spending enough effort for preparation directly results in more detected defects.

Apart from more insight into the causal relationships, the findings presented in this paper provide a baseline against which to compare changes to the review process. So far, the review meeting at PRC-ON is conducted as a so-called 'face-to-face meeting', by default. Due to the increasing integration of PRC-ON into international development activities, some reviewers are connected by means of a telephone conferencing system to participate in the review meeting. Since a review object may be made accessible through the world-wide Lucent Intranet, the cost-effectiveness of a single review does not seem to be affected by such 'distributed reviews'. However, since meetings in general represent a major cost factor for reviews, non-meeting based approaches are a fruitful area for further investigation.

Finally the chosen analysis strategy based on path analysis was appealing because it allows for a systematic testing of a theoretical model, which in turn allows one to integrate existing work in a unified framework. Other researchers may use the model to drive their analysis. This kind of strategy allows the software engineering community to build theories on the conditions where software reviews are most beneficial.

7. References

[1] V. R. Basili. Evolving and Packaging Reading Technologies. Journal of Systems and Software, 38(1), July 1997.

[2] W. D. Berry and S. Feldman. Multiple regression in practice. Sage Publication, 1985.

[3] K. V. Bourgeois. Process Insights from a Large-Scale Software Inspections Data Analysis. Cross Talk, The Journal of Defense Software Engineering, pages 17–23, oct. 1996.

[4] D. A. Christenson, H. T. Steel, and A. J. Lamperez. Statistical quality control applied to code inspections. IEEE Journal Selected Areas in Communication, 8(2):196–200, February 1990.

[5] J. Cohen and P. Cohen. Applied Multiple Regression/ Correlation Analysis for the Behavioral Sciences. Lawrence Erlbaum Associates, Inc., Publishers, 1983.

[6] C. Ebert, T. Liedtke, and E. Baisch. Software Measurement - Current Trends in Research and Practice, chapter Improving Reliability of Large Software Systems, pages 209–228. Deutscher Universitaets Verlag, Gabler Edition Wissenschaft Edition, 1999.

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

[8] T. Gilb and D. Graham. Software Inspection. Addison-Wesley Publishing Company, 1993.

[9] L. Hatton. Reexamining the Fault Density-Component Size Connection. IEEE Software, 14(2):89–96, 1997.

[10] P. M. Johnson and D. Tjahjono. Does Every Inspection Really Need a Meeting. Journal of Empirical Software Engineering 3(1):9–35, 1998.

[11] J. C. Kelly, Joseph S. Sherif, and J. Hops. An analysis of defect densities found during software inspections. Journal of Systems and Software, 17:111–117, 1992.

[12] M. Leszak and W. Kammerer. Review Process Improvement in Transmission Network Development. In Proceedings of the First Conference on Quality Engineering in Software Technology, pages 20–28, 1997.

[13] R. Madachy, L. Little, and S. Fan. Analysis of a successful Inspection Program. In 18th Ann. NASA Software Engineering Laboratory Workshop, pages 176–198. NASA, November 1993.

[14] E. J. Pedhazur. Multiple Regression in Behavioral Research. Hartcourt Brace College Publishers, second edition, 1982.

[15] A. Porter, H. Siy, A. Mockus, and L. Votta. Understanding the Sources of Variation in Software Inspections. ACM Transactions on Software

Engineering and Methodology, 7(1):41–79, January 1998.

[16] A. Porter, H. Siy, C. Toman, and L. Votta. An Experiment to Assess the Cost-Benefits of Code Inspections in Large Scale Software Development. IEEE Transactions on Software Engineering, 23(6):329–346, June 1997.

[17] A. Porter and L. Votta. What Makes Inspections Work? IEEE Software, 14(6):99–102, November 1997.

[18] T. Raz and A. T. Yaung. Factors affecting design inspection effectiveness in software development. Information and Software Technology, 39:297–305, 1997.

[19] R. Retherford and M-K. Choe. Statistical Models for Causal Analysis. John Wiley & Sons Inc, 1993.

[20] G. W. Russell. Experience with Inspection in Ultralarge-Scale Developments. IEEE Software, 8(1):25–31, January 1991.

[21] C. B. Seamann and V.R. Basili. Communication and Organization: An Empirical Study of Discussion in Inspection Meetings. IEEE Transactions on Software Engineering, 24(6):559-572, 1998.

[22] H. Siy and L. Votta. Does The Modern Code Inspection Have Value? Submitted for Publication.

[23] L. Votta. Does Every Inspection Need a Meeting? ACM Software Engineering Notes, 18(5):107–114, December 1993.

[24] E. F. Weller. Lessons from Three Years of Inspection Data. IEEE Software, 10(5):38–45, September 1993.