An Apporach to Improving Software Inspections Performance

Andre L. Ferreira, Ricardo J. Machado
Dept. de Sistemas de Informação, Universidade do Minho
{andre.ferreira,rmac}@dsi.uminho.pt

Lino Costa
Dept. de Produção e Sistemas, Universidade do Minho
lac@dps.uminho.pt

José G. Silva and Rui F. Batista
Critical Software S.A. Portugal
{jsilva,rui.f.batista}@criticalsoftware.com

Mark C. Paulk
Institute for Software Research
Carnegie Mellon University, USA
mcp@cs.cmu.edu

Abstract— **Software inspections allow finding and removing defects close to their point of injection and are considered a cheap and effective way to detect and remove defects. A lot of research work has focused on understanding the sources of variability and improving software inspections performance. In this paper we studied the impact of inspection review rate in process performance. The study was carried out in an industrial context effort of bridging the gap from CMMI level 3 to level 5. We supported a decision for process change and improvement based on statistical significant information. Study results led us to conclude that review rate is an important factor affecting code inspections performance and that the applicability of statistical methods was useful in modeling and predicting process performance.**

Software inspections; Software Process Improvement

I. INTRODUCTION

It is generally accepted that quality in software remains a challenge. A major quality issue with software is that defects are a byproduct of the complex development process and the ability to develop defect free software remains a big challenge for the software community. It is possible to improve the quality of software product by injecting fewer defects or by identifying and removing defects injected. Software testing is an activity for defect identification and removal of these defects. Testing can be static if a simple examination of the software artifact is performed to find problems or dynamic if an actual run of the software is required [1].A form of static testing is software inspections. An inspection is characterized *"as a systematic approach to examine a product in detail, using a predefined sequence of steps to determine if the product is fit for its intended use"* [2]. Software inspections require that a person, usually called a reviewer, spends and amount of time analyzing the product to find defects. Whatever the structure/steps of the inspection process assumes, there is an amount of time spent by a reviewer inspecting the product and a corresponding number of defects found resulting from the review.

It is recognized that software inspections are a simple, and cost effective approach to detect and eliminate defects [3].They require less training when compared to other defect detection techniques. Inspections are cost effective in the perspective that detecting and fixing defects at earlier phases of development requires less effort when compared to finding and fixing these same defects at later phases of development. Moreover, inspections provide means to improve maintainability of software by allowing to detect certain types of defects that are not detectable using other defect detection techniques [4].An example are evolvability defects found in a ratio between 5:1 and 3:1 to functional defects that otherwise would never be identified [5].

It is also general accepted that performance of software inspections is affected by several factors. Inspection performance is associated with the effort spent to carry out the process and/or the number of defects found.

A wide set of empirical studies and new approaches have been proposed to understand and improve the inspection process since it was introduced by Fagan [6].Sources of process variability range from structure (how steps of the inspection are organized), inspection inputs(reviewer ability and product quality) techniques (applied to defect detection that define how each step is carried out), context and tool support [7]. Most empirical studies try to assess the impact of specific process settings on performance. Some sources of variation are: absence or presence of inspection meetings, experience of the reviewers, initial code quality, number of reviewers participating in the inspection, time spent to detect defects and the rate at which products are inspected. Despite the efforts, a general 'theory' that combines this sources of variability into a comprehensive set it still to be unveiled [8].

Review rate seems to be an important factor affecting inspection performance. High review rates have been conceptually and empirically associated to a decrease in inspections effectiveness [9]. When considering code as the inspected artifact, review rate establishes the number of lines of code (LOC) each reviewer reads per hour to find defects. Recommended rates that maximize the number of defects found are around 125 LOC/hour with a fast decline for rates above 200LOC/hour [10, 11]. However, limited investigations are available on the subject of finding the optimal rate to perform code inspections in industrial settings [9].

Recently, Critical Software S.A., a Portuguese software house was assessed for CMMI-Dev [12] maturity level 5.Under the scope of the Organizational Innovation and Deployment process area implementation, a goal was set to improve the inspection process. An objective was defined to understand process performance by building a process performance model.

We were required to base a decision to change and improve the process on a quantitative estimate of improvement.

This paper describes the approach taken to build this process performance model for the inspection process and how it was used to support a decision on how to improve it. From the work we concluded that review rate is an important factor in code inspection performance, reinforcing previous research on the subject. We obtained a statistical significant improvement in the number of defects found when controlling review rate. Despite the improvement in the average number of defects found, variability of results also increased. Extra analysis allowed us to identify that reviewers contributed to this variation with statistical significance.

The following sections are organized as follows: Section 2 describes relevant previous research work on software inspections. Section 3 describes and discusses the approach and data used in the study. Section 4 summarizes results and elaborates on lessons learned from the study.

II. State of the Art

Software inspections were introduced by Fagan in 1976 [6]. The goal of performing inspections is to improve the software product quality by finding and removing product defects. Main advantage of inspections when compared to other verification and validation activities is that defect identification and removal occurs closely to the point of injection, thus effort associated with finding and fixing defects is reduced [6].

An inspection is defined as a sequence of process steps or operations. The original proposal by Fagan considers five steps, namely: *overview, preparation, inspection meeting, rework and follow-up* [6]. Since their introduction, inspections were subject of considerable research [8].The literature makes available a set of experiments designed to identify and quantify sources of process variability with the goal of improving inspection performance. Performance can be characterized as process effectiveness and efficiency. Effectiveness of inspection is the capability of the inspection process to detect all existing defects in the software artifact. Efficiency is concerned with the effort spent in finding those defects.

Sources of variability can be associated with process structure, techniques used and process inputs. Studies focusing on process structure often re-define or remove process steps from the initial structure proposed by Fagan. Studies on process inputs involve controlling inputs by engineering or tightly controlling their attributes. A correlation is later established between these attributes and the resulting process performance.

Process inputs account for sources of variation that change across inspections; these are related with who inspects and what is inspected. Process structure and techniques are associated with how the steps are organized and how they are carried out, respectively.

A. Process structute as source of variation

Since the introduction of inspections by Fagan, a few approaches have been proposed by several authors. Those focus mainly on re-definition of the process steps. Examples are Active Design Reviews [13], Phased Inspections [14] and N-fold inspections [15]. These methods rely mainly on the argument that several persons focusing on special inspection techniques are more effective in finding defects than a single large team with no special techniques.

A synthesis on the subject of verification and validation is available in [16]. The term peer review is used to name any activity of reviewing software products created during software development. Peer reviews are categorized as deck-checks, walkthrough or inspections. The degree of formality may vary in deck-checks and walkthroughs. Inspections are strictly formal reviews and their applicability varies according to the state of the product. The previously mentioned approaches are examples of formal reviews.

A formal review or inspection according to [16] follows the typical Fagan approach with a slight relevant change. The preparation step is used both for understanding and inspection. This simple change motivated, in the past, most empirical studies to evaluate the relevance of conducting the meeting inspection step. If we consider a cost efficiency analysis, it is argued that meetings require too much effort for the additional defects found.

An example is a controlled experiment by Johnson and Tjahjono [17]. They analyzed the impact of executing inspections with and without the inspection meeting step. They analyzed the impact on the following variables: *total defects, effort, false positive defects, duplicates* and *synergism*. They were unable to find significant differences in the total number of defects found when comparing meeting-based with meeting-less-based inspections (the meeting is eliminated and inspection occurs only in the preparation step). Conversely, the meeting-based required more *total effort* and *effort per defect* but resulted in significant less *false positives defects (*defects that were considered by the rework responsible as not true defects). Also, *synergism* (interaction between reviewers in meetings) as a result of meeting-based inspections resulted in 30% of total defects identified. Meeting-less-based inspections resulted in more issues but one third of these issues were *duplicates* (found by more than one reviewer). Although results in what concerns productivity point in favor of meeting-less-based inspections, as cost per defect is lower, meetings allow participants to share review experiences, obtain insight into overall effectiveness of review, gain additional insight into the work product and its quality and finally it fosters collective ownership and responsibility for the review outcome.

McCarthy and Porter also suggest that meetings are not necessarily essential to successful inspections [18].They carried out the study to clarify previous studies on the matter that apparently reported meeting gains of 33% on defects found [19]. They measured total defects applying different inspection structures. More defects were identified with meeting-less-based approaches in a context where artifacts inspected were requirements specifications.

Another structural factor subject of study is the optimal number of reviewers performing inspections. In this matter a study by Porter *et.al* [20] compared the performance of two and four element teams as single inspections. No improved performance in effectiveness was attained with four reviewers when compared with two reviewers. Single inspections were less effective than two elements inspection, but the difference was small.

Kantorowits *et. al.*[21] explored the use of an estimator to determine inspections team size. The estimator considers as parameters a characterization of detection ability and code domain knowledge of the reviewers to obtain a desired defect detection probability. The characterization of detection ability is a function of the individuals performing the inspections, type of artifact inspected and the method used for inspection. Code domain knowledge is specified as a continuous value between 0 and 1 and characterizes the limitation resulting from lack of knowledge in the domains by the reviewers in detecting defects. The study showed that values delivered by the estimator are close to the observed values from a controlled experiment. The estimator delivers the required number of reviewers to attain the desired defect detection. A reference to no more than seven reviews per meeting is suggested in [16].

Other aspects of process structure are preparation time and meeting duration. Is was been suggested that an increase in the preparation time correlated with the number of defects found [20]. Meeting duration also correlated with the number of defects found and meetings should occur no more than twice a day and duration should not exceed 120 minutes.

Another structural related factor is the review rate at which inspection are performed. It relates product size (number of lines of code, requirements pages, etc) with the time each reviewer takes to inspect the product. High reviews rates are associated with a decrease in review effectiveness [9]. Recommended rates for preparation are around 100 and 125 LOC/hour with fast decline for higher above 200 LOC/hour [9-11].

B. *Techniques as sources of variation*

Hatton *et. al.* conducted a rigorous experiment to assess the impact on inspection performance of using checklists [22]. The result was inconclusive on whether the checklist improved the number of defects found when enforcing a code review rate. No statistical significance was obtained from the results of the experiment. However the checklist focused only in one type of defect and code reviewed was relatively short. Reference to the successful use of checklists to aid the inspection process are described in [23].

A technique to improve the effectiveness of the inspection meeting is explored by Vitharana and Ramamurthy in [24]. Through a controlled experiment, they studied the influence of anonymity in meeting/team based inspections. Anonymity implies that elements of the inspection team do not know the identity of participating elements. They observed the effect of anonymity in an experiment with a control group, by controlling the following variables: effectiveness (total defects), efficiency (less time) and reviewer attitude (freely express their tasks centered comments and views). They used two distinct groups with different background experiences and samples of code with different levels of complexity. The results showed that anonymity had no impact on efficiency and a significant impact on effectiveness occurred only when the code sample for inspection was classified as more complex. Anonymity also favored inspector attitude towards the inspection.

C. *Inputs as source of variation*

Important sources of variability are the artifacts to be inspected and the reviewer inspecting the product. It is expectable that a product with a high number of initial defects be associated with high number of defects found. This association depends also on the reviewers' ability to find defects. Reviewers less able or less experienced are expected to find fewer defects.

Nair and Suma conducted an empirical experiment to study the effectiveness of the inspection process. They observed project data from several leading service based and product-based software companies rated at level CMM level 5 [25]. Two metrics were considered to quantify the capability of the inspection process in capturing defects within the constraints of parameters affecting inspections. The first, characterized as people metric is the inspection performance metric (IPM) that considers the number of defects caught in the inspection process (NI) over the inspection effort (IE). The second is depth of inspection (DI) characterized as a process metric, considers NI over the total number of defects (TD), where TD is NI plus the number of defects caught in the testing process. DI is characterized as a measure of effectiveness of inspection, defect prevention metric, quality metric and a measure of the ability of the inspection process in reducing the test effort. From the observed data they concluded that major sources of variability on effectiveness of inspections are the number, experience and skill of the inspectors but also preparation and inspection time. The IPM and DI are proposed as benchmarking tools to improve industry defect management practices. In another study Vitharana, and Ramamurthy also concluded that more experience has a significant impact in efficiency and effectiveness of inspections [24].

In summary, it is not clear the degree to which process structure impacts effectiveness, but the impact seems to be small. Efficiency seems to be negatively affected by performing the meeting step. It is plausible that meetings improve the number of detected defects but require substantially more effort. Effectiveness also seems too improve when inspections are performed individually when compared with in a team based approach. Scenario-based detection techniques seem to be more effective than *ad-hoc* or checklist approaches and process inputs seam to explain more variation than structural factors [8]. Despite the effort for improving inspections, the relevant problem seems to be the wide adoption of inspections spite the overwhelming evidences of their benefits [26, 27].

III. IMPROVING INSPECTIONPROCESS PERFORMANCE

This section documents the approach taken to improve the performance of the inspection process. The project was carried out in a Portuguese software house, Critical Software S.A (CSW). The study was performed in the context of a program to implement the necessary practices to bridge the gap of a

CMMI maturity level 3 to a maturity level 5. A goal was set to improve code inspection process performance as an Organizational Innovation and Development process area project. To achieve this goal we followed a typical Define, Measure, Analyze, Improve and Control model for process improvement [16], by:

A. Define the problem and goals for process improvement

The main goal motivating project efforts was to select and deploy an improvement for the inspection process. This goal was aligned with organization business quality objectives for product quality improvement. The challenge was to understand code inspection performance and obtain a strong statistical estimate of improvement to support the decision for implementing a change in the process. This required a preliminary quantifiable understanding or model of process performance.

We assumed that our ability to find defects was not in an optimal value and could be improved. This implied we were looking to improve process defect detection capability. We considered defects as '*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*' [28] and used an Orthogonal Defect Classification [29] based checklist to guide defect classification.

B. Measure and collection of process performance data

As a formerly CMMI level 3 rated organization, CSW has a defined and disseminated a process for verification and validation of software products. The process includes a procedure for performing code inspections monitored by a set of metrics. The process structure follows similar steps as described in [16], namely:

Preparation (P), each reviewer inspects the code. A code review checklist is used in the preparation step to help the reviewer perform the inspection. A list of defects is expected as a result of this step.

Review Meeting (M), in the review meeting, a walkthrough reading of the code sample is performed to collect and discuss each defect identified in the preparation step. The author and reviewers participate in the meeting. Additional defects are recorded if identified as result of the group interactions. A full reading of the code may be performed but it's not mandatory.

Rework (R), the author receives a list of defects that must be removed. In a later step, the removal is validated by the inspection responsible.

Entity	Attribute	Metric
Code	Size	Lines of code (LOC)
Code	Language	Categorical
Code	Pre-inspection unit testing	Yes/No

Table 1 - Process Input Entities

Based on these steps, two types of reviews are possible. A first one includes preparation, review meeting and rework (P-M-R). In the second, only preparation and rework (P-R) are performed. The available set of metrics, previously collected, enabled us to characterize the entities participating in the inspection process as follows. Table 1 depicts input related entities. Table 2 depicts process structural-related entities. Each attribute has an associated base measure.

Entity	Attribute	Metric
Preparation session	Duration	Hours
Review meeting	Duration	Hours
Review meeting	Review team size	Number

Table 2 - Process Structural Entities

Based on this set of process related metrics, we considered code inspection review rate as the primary factor affecting process performance and modeled inspection effectiveness as a function of code inspection rate.

$$CodeRevEffect \left[\frac{defects}{loc}\right] = f(CodeRevRate)\left[\frac{loc}{hour}\right] \quad (1)$$

Code review effectiveness is a measure of defects found by reviewers normalized by the size of code inspected. Review rate is the derived measure that relates code size and preparation session duration in number of lines of code per hour of review performed. The decision to consider a single factor in (1) to model code inspection process was support by two reasons: firstly, available data from past inspections limited the number of variables that could be considered for modeling process performance. We needed past process metrics and collecting extra data to characterize previous inspections did not present itself as a viable option as it required collecting information on past events and could not present itself as a very reliable approach to collect quality data. We made a choice to work with existing data.

Secondly, data available associated to the majority of process entities had limited variation. Concerning process structure, all inspection followed a P-M-R process structure. The range of values for review team size varied from 3 to 4 reviewers and review meeting duration were around 120 minutes with no significant variation. Conversely, review rate variability was considerable. No control was enforced by the defined process. It was up to the reviewer to inspect the code at the desired rate. This provided a good base for studying the impact of review rate on defect density. Additionally, review rate has been considered, based on empirical evidence, by several authors as an important non-negligible factor impacting the number of defect found.

With this decision we assumed that review rate was the main justification for process performance variation in the specific context we were performing inspections. The data used to build the model was obtained from code inspections performed by professional developers participating in a total of three projects. These were characterized as being representative of the typical software development projects at CSW. A sample of three projects was chosen. These used C programming language and were characterized by having atypical team size and project length.

Inspection data was collected from inspections following a P-M-R inspection process structure. Inspection teams varied from 3 to 4 reviewers. The reviewers spent an average of 90 minutes in the preparation session. A single inspection meeting lasted no more than 120 minutes. A sample of 45 code inspections was considered to build the model, after a data set reduction to eliminate deficient quality records and outliers. Prior to inspection, code was subject to static analysis using a compiler. Reviewers were members of the development team and inspections were first time inspections (the code was never reviewed before).

For collection, data sheet templates were used to gather inspection data. All reviewers received from the inspection moderator a code sample to be reviewed. Each reviewer registered defects, typically in a printout of the code sample. In the review meeting the moderator registered the time each reviewer spent on the preparation session, the final total number of defects found and code size. The inspection meeting duration was also registered.

C. Analyze performance and consider possible improvements

With inspection data available the analysis step focused on finding an association between review rate and defect density. Additionally, we needed to build a model to estimate defect density based on review rate data. A regression analysis was used with the goal of characterizing this possible association. Figure 1 depicts a scatter plot of how defect density (y-axis) varies with inspection code rate (x-axis). Since data is from real project inspections the values for defect density are masked. Based on the scatter plot information we started by considering a linear association between variables. Firstly, we considered the conditions of applicability of regression analysis. Using the Kolmogorov-Smirnov non-parametric test [30] the normality of distributions of both variables was confirmed. Table 3 lists the models considered to explain the relation between variables and the resulting R-square.

Model	Formula	R-square	Sig
Linear	$Y(t)=b_0+b_1t$	0.317	0.000
Inverse	$Y(t)=b_0+b_1/t$	0.583	0.000
Quadratic	$Y(t)=b_0+b_1t+b_2t^2$	0.485	0.000
Cubic	$Y(t)=b_0+b_1t+b_2t^2+b_3t^3$	0.569	0.000
Power	$Y(t)=b_0t^{b1}$	0.093	0.044
Exponential	$Y(t)=b_0e^{b1t}$	0.061	0.170

Table 3 - Considered fit models

The Linear, Inverse, Quadratic, Cubic and Power regression models are statistically significant (p-values<0.05). The inverse model has the highest R-square with a value of 0.583, a Fisher test statistic of F(1;42) = 58.676 for the Analysis of Variance (ANOVA) and a p-value=0.000<0.05 (Table 4). The Inverse model explains 58.3% of the variability of the dependent variable. The data and models curve fits are depicted in the scatter plot (see Figure 1).With the objective to obtain a better fit with a higher R-square we tested a curve fit without the constant b_0. We obtained a better fit with an improved R-square of 0.752. The inverse model without constant is significant (p-value=0.000<0.05) with a Fisher test statistic value of F(1;43)=130.218 and with a higher R-square (Table 5). The model is given by $Y(t)= b_1/t$ and it is linear in terms of their parameters.

Equation	Model Summary	
	R-square	Sig
Inverse	0.583	0.000

Table 4–Inverse model summary

We tested an additional non linear model based on the inverse relation: $Y(t)=b_0/t^{b1}$. The R-square obtained was 0.61 (inferior to the inverse model without constant).

Equation	Model Summary	
	R-square	Sig
Inverse	0.752	0.000

Table 5 - Inverse model without constant summary

Thus, the model with higher prediction power is the Inverse model without constant. Having an acceptable R-square we considered the inverse model as the best predictor of process performance, considering defect density as a function of preparation review rate.

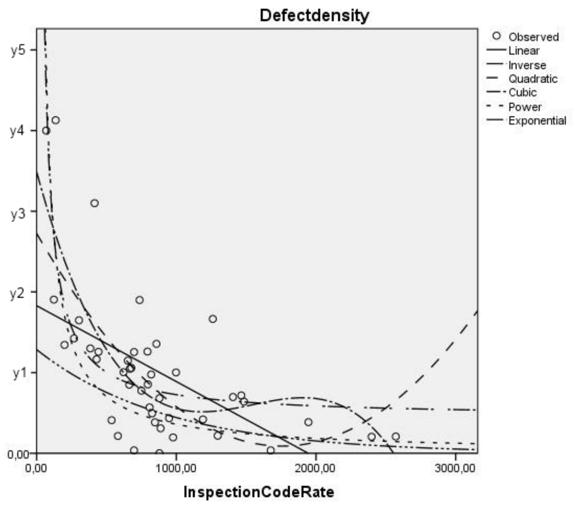

Figure 1 - Data and linear models considered

D. Change the process definition based on quantitative data.

As we mentioned earlier, our goal was to change the inspection process based on a quantitative estimate of improvement. Looking at the adopted model (Table 5 and Figure 2) it is possible to see that a wide range of review rates were being used by reviewers. The average value was about 800LOC/hour and defect density declined considerably for high review rates.

We considered then a change to the process. A new review rate was to be used for performing inspections in order to improve the number of defects found. Based on literature recommendations that argue a maximum of 200 LOC/hour we used the model to estimate the expected defect density for this review rate. We compared it with the project sample average defect density. If the model was accurate, reducing the review rate would provide a 70% increase in defect density.

Based on this information, we decided to carry out a pilot program where inspections would be performed with a controlled review rate. Firstly, we wanted to assess if in fact the defect density would improve and secondly, if the improvement was achieved, in which percentage did it occur.

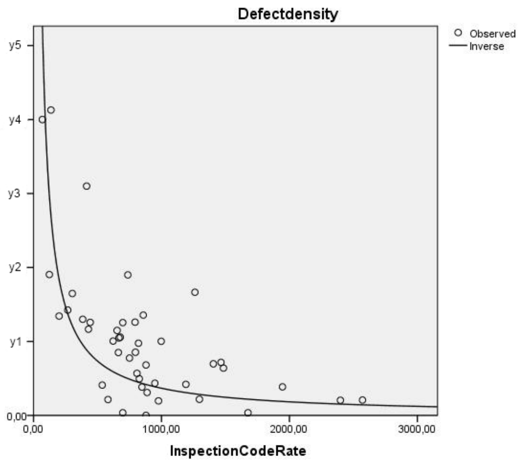

Figure 2 – Code inspection performance model

E. Assess the impact of reducing review rate

A set of projects for the pilot program were chosen to perform code inspections using the specific review rate. A new project context was in place for the pilot program. The set of projects available to perform code inspections limited the possibility to have a similar context to the sample projects that originated the data for the performance model. We could not control all the dependent variables. This fact was a challenge to the predicting power of the model.

Inspections were now performed in projects using JAVA programming language. The reviewers performing inspections also changed and the structure used to perform the inspections was the P-R approach (the review meeting did not occur). The inspections were also performed by a single reviewer.

Based on the literature, these changes could impact the ability to find more defects and by that, put at risk the improvement in process performance. In what concerns the meeting step, it has been argued that removing the meeting step does not impact significantly the ability to find more defects [17].Thus, an increase in the number of defects found was not expectable by removing the meeting step.

A performance decrease was expectable by using a single reviewer instead of a team of reviewers. Intuitively, a team of 3 or 4 reviewers is able to find more defects than a single reviewer. The added experience or ability of each reviewer may improve defect detection. Although, previous studies concluded that a decrease in performance for single reviewer inspections may be negligible when compared with two and four element teams [20].

Some factors with uncontrolled influence in defect density were: programming language used, impact of the reviewers performing the inspections and possible variation in initial code quality.

The pilot inspections followed a similar collection procedure as described in sub-section B except that the reviewers had a review rate criterion to be met. A review rate between 200 and 250 LOC/hour as recommended to the reviewers. Additionally, a single preparation session should not take longer than 120 minutes.

Defects were registered in data sheet templates and were sent directly to the author for rework. During this pilot study extra context information was collected to investigate possible influence of the unknown factors. Motivated by the literature studies concerning the impact of the reviewer, we decided to collect information about *ability* and *inspection experience* of reviewers. *Ability* is defined here as the number of years developing in the programming language used in the code and *inspection experience* as the number of formal reviews performed in the past. A total of 39 inspections were carried out by a pool of three reviewers in a context of four different projects.

1) Inspection review rate impact on defect density

The scatter plot of Figure 3 depicts the results for defect density for the pilot inspections. The plot depicts the same relation of the performance model in sub-section C relating defect density and inspection review rate.

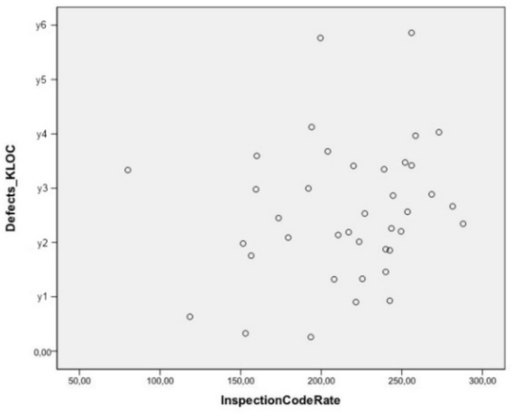

Figure 3 - Scatter plot for new inspection setting

The Pearson correlation coefficient was computed to measure variables association. With the review rate controlled no significant correlation between the variables is evident (p-value=0.233>0.05).

The average review rate changed from 800LOC/hour to 215 LOC/hour with a standard deviation of 46. The average value for defect density significantly improved. The box plot (Figure 4) depicts the variation in average defect density values for both scenarios, prior (1.00) and after (2.00) the controlled review rate.

A statistical t-test for independent samples was used to evaluate the significance of this variation (Table 6). The test indicates that defect density increased significantly, t(68.167)=-6.306 and p-value=0.000 <0.05.

Analyzing the box plot we suspected also an increase in the variability of results. A Levene test was used to assess equality of variances for both samples (Table 6). The test provided a p-value=0.033<0.05 therefore, the variances are significantly different. Variability has increased in the review rate controlled scenario. We also checked to which extension the model delivered a reliable estimate of improvement. If the resulting defect density value for the pilot inspections was within the 95% percent range of the value predicted by the model, we would accept the model as a reasonable approximation of real process performance.

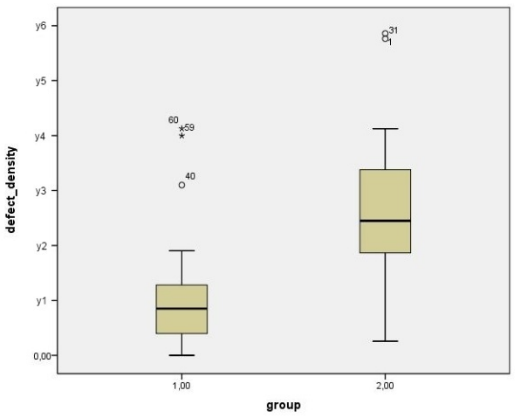

Figure 4 – Defect density by scenario

Variances	Levene Test		t-test for Equality of Means		
	F	Sig.	t	df	Sig. (2-tailed)
Equal	4.692	0.033	-6.430	0.000	0.000
Unequal			-6.306	0.000	0.000

Table 6 - Test for equality of means and variances

We used the average review rate from the pilot (215LOC/hour) and used the performance model to get the defect density estimate for that specific rate. A 95% percent confidence interval was computed for the resulting defect density. The top limit for improvement acceptable by the model implied a 165% improvement in defect density. The obtained value for defect density by the pilot was 142%.

Thus, the value obtained in the pilot study is within the 95% confidence interval of the estimated value by the model, leading us to conclude that the model provided an acceptable prediction of process performance.

2) Understanding the increase in process variability

In order to try to understand the increase in the variability, we isolated two factors with the expectation that, at least, one of them would provide some explanation for such variability. We began by checking if variability could be explained by variation in initial code quality. We considered projects to study possible differences in initial code quality. For the second source of variation we considered the reviewers impact. One of the reviewers was a senior developer, the others were juniors. This classification translated their ability as developers and experience as reviewers. The senior status evidences both more years using the programming language and more experience as a reviewer.

To test the impact of code quality, a box plot of defect density by project is depicted in Figure 5. Applying an one-way ANOVA to assess significance in the two or more samples, the following result is obtained: F(3,35)=1.197 and p-value=0.325>0.05 (Table 7). This indicates that differences in average defect density are not significant between projects.

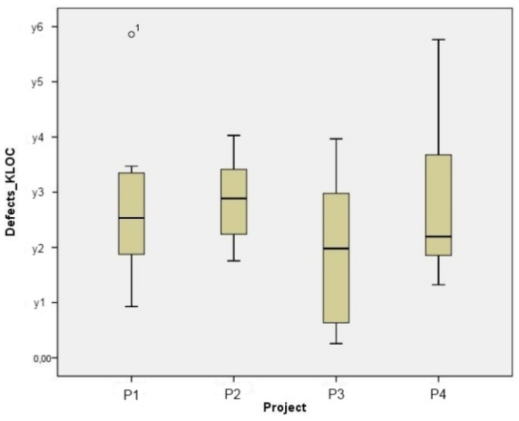

Figure 5 - Defect density by project

Groups	Sum of Squares	df	Mean Square	F	Sig
Between	2264,417	3	754,806	1.197	0.325
Within	22076,897	35	630,768		
Total	24341,314	38			

Table 7 - ANOVA test for project means

The box plot showing the impact on defect density by reviewer is depicted in Figure. We applied the ANOVA test and the following result is obtained(2,36)=4.620, p-value=0.016<0.05 (Table 8). This indicates the differences in the average defect density by reviewers are statistically significant.

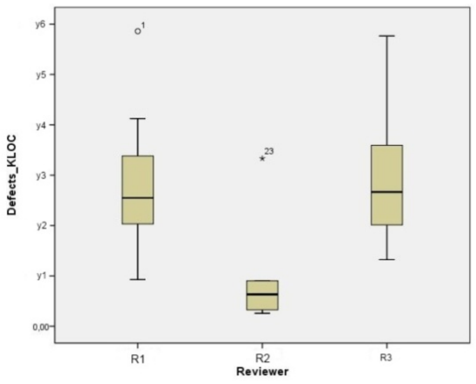

Figure 6 - Defect density by reviewer

Groups	Sum of Squares	df	Mean Square	F	Sig
Between	4971,348	3	2485,675	4.620	0.016
Within	19369,965	36	539,055		
Total	24341,314	38			

Table 8 - ANOVA test for reviewer means

A deeper analysis on the reviewer's academic background allowed finding that R3 had above average performance as a student developer. This fact may justify the variation in performance of the reviewer and as result a source of significant performance variability in the pilot study.

Additionally, an interesting finding was that there was a difference in reviewer experience and programming experience between reviewer R1 and R3 but the final average defect density was similar.

IV. Conclusion

This paper documents an approach to improve the performance of an inspection process. We were required to support a decision to change the process on a quantitative estimate of improvement. A model of the inspection process performance as a function of inspection review rate was defined. We applied statistical methods (linear and non-linear regression) to understand and build a real process performance model. Based on the estimate delivered by the adopted model we decided to control the review rate by reducing it. We tested this process change experimentally and obtained an improvement in process performance. We also validated statistically the predictive power of the estimate with the experimental result.

In an industrial setting is not always possible to have a strict control of the environment. We had limited control in some process variables that were important to validate with strong statistical reasoning the impact of the review rate in process performance. The limitation came mainly due lack of 'perfect match' projects to replicate the same conditions used when the first set of inspection data was collected. Additionally, data volume in terms of number of reviewers, number of projects and number of inspections limits the ability to generalize on the study findings, concerning a deeper understanding of sources of variation for the general inspection process. Even so, we carried out the experiment in a context that was useful to understand the impact of review rate in process performance and provided additional evidence of previous research that considered the review rate an important factor in inspections performance.

References

[1] L. Hatton. Testing the Value of Checklists in Code Inspections. In *Software, IEEE,* vol. 25, no. 4, pp. 82 - 88, 2008.

[2] D. L. Parnas and M. Lawford. Inspection's role in software quality assurance. In *Software, IEEE,* vol. 20, 2003.

[3] O. Laitenberger. Studying the effects of code inspection and structural testing on software quality. In *Software Reliability Engineering*, Proceedings. The Ninth International Symposium on, pp. 237-246, 1998.

[4] H. Siy, and L. Votta. Does the modern code inspection have value? In *Software Maintenance, 2001. Proceedings. IEEE International Conference on*, 2001.

[5] M. V. Mantyla, and C. Lassenius. What Types of Defects Are Really Discovered in Code Reviews. In *Software Engineering, IEEE Transactions on,* vol. 35, no. 3, pp. 430-448, 2009.

[6] M. E. Fagan. Design and Code inspections to reduce errors in program development. In *IBM Systems Journal 15* pp. 182-211, 1976.

[7] D. E. Perry, A. Porter, M. W. Wade. Reducing inspection interval in large-scale software development. In *Software Engineering, IEEE Transactions on,* vol. 28, no. 7, pp. 695-705, 2002.

[8] A. Porter, and L. Votta. What makes inspections work? In *Software, IEEE,* vol. 14, no. 6, pp. 99-102, 1997.

[9] C. F. Kemerer, and M. C. Paulk. The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data. In *Software Engineering, IEEE Transactions on,* vol. 35, no. 4, pp. 534-550, 2009.

[10] M. E. Fagan. Advances in software inspections. In *IEEE Trans. Softw. Eng.,* vol. 12, no. 7, pp. 744-751, 1986.

[11] E. F. Weller. Lessons from three years of inspection data [software development]. In *Software, IEEE,* vol. 10, no. 5, pp. 38-45, 1993.

[12] M. B. Chissis, and M. Konrad. CMMI for Development, Version 1.2. *Addison-Wesley*, 2006.

[13] D. L. Parnas, and D. M. Weiss. Active design reviews: principles and practices. In *Proceedings of the 8th international conference on Software Engineering*, London, England, 1985.

[14] J. C. Knight, and E. A. Myers. In an improved inspection technique. In *Commun. ACM,* vol. 36, no. 11, pp. 51-61, 1993.

[15] G. M. Schneider, J. Martin, and W. T. Tsai. An experimental study of fault detection in user requirements documents. In *ACM Trans. Softw. Eng. Methodol.,* vol. 1, no. 2, 1992.

[16] L. Westfall. *The Certified Software Quality Engineer Handbook*: ASQ Quality Press, 2009.

[17] P. M. Johnson, and D. Tjahjono. Does Every Inspection Really Need a Meeting? In *Empirical Software Engineering,* vol. 3, no. 1, pp. 9-35, 1998.

[18] P. McCarthy, A. Porter, H. Siy *et al.* An experiment to assess cost-benefits of inspection meetings and their alternatives: a pilot study. In *Proceedings of the 3rd International Symposium on Software Metrics: From Measurement to Empirical Results*, 1996.

[19] A. Porter, H. Siy, C. A. Toman *et al.* An experiment to assess the cost-benefits of code inspections in large scale software development. In *SIGSOFT Softw. Eng. Notes,* vol. 20, no. 4, pp. 92-103, 1995.

[20] A. A. Porter, H. P. Siy, C. A. Toman *et al.* An experiment to assess the cost-benefits of code inspections in large scale software development. In *Software Engineering, IEEE Transactions on,* vol. 23, no. 6, pp. 329-346, 1997.

[21] E. Kantorowitz, T. Kuflik, A. Raginsky. Estimating the Required Code Inspection Team Size. In *IEEE International Conference on Software-Science, Technology & Engineering* (SwSTE'07), 2007.

[22] L. Hatton. Testing the Value of Checklists in Code Inspections. In *Software, IEEE,* vol. 25, no. 4, pp. 82-88, 2008.

[23] J. R. de Almeida, Jr., J. B. Camargo, Jr., B. A. Basseto *et al.* Best practices in code inspection for safety-critical software. In *Software, IEEE,* vol. 20, no. 3, pp. 56-63, 2003.

[24] P. Vitharana, and K. Ramamurthy. Computer-mediated group support, anonymity, and the software inspection process: an empirical investigation. In *Software Engineering, IEEE Transactions on,* vol. 29, no. 2, pp. 167-180, 2003.

[25] G. Nair, and Suma, V. Impact Analysis of the Inspection Process for Effective Defect Management in Software Development. In *Software Quality Professional,* vol. 12, no. 2, 2010.

[26] C. Denger, and F. Shull. A Practical Approach for Quality-Driven Inspections. In *Software, IEEE,* vol. 24, no. 2, 2007.

[27] J. Remillard. Source code review systems. In *Software, IEEE,* vol. 22, no. 1, pp. 74 - 77, Jan 1, 2005.

[28] IEEE. Standard Classification for Software Anomalies. 2009.

[29] R. Chillarege, I. S. Bhandari, J. K. Chaar *et al.* Orthogonal defect classification-a concept for in-process measurements. In *Software Engineering, IEEE Transactions on,* vol. 18, no. 11, pp. 943-956, 1992.

[30] A. Field. *Discovering statistics using SPSS.* Sage, 2009.