

On the Influence of Practitioners' Expertise in Component Based Software Reviews

Miguel Goulão¹, Fernando Brito e Abreu¹

¹ QUASAR Research Group, Centro de Informática e Tecnologias de Informação
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Portugal
{miguel.goulao, fba}@di.fct.unl.pt
<http://ctp.di.fct.unl.pt/QUASAR/>

Abstract. *Objective:* To assess the influence of practitioners' expertise in code inspection of software components. *Method:* Subjects expertise was determined based on their independently assessed academic record. Inspection outcome was represented by the diversity of defects found at two levels of abstraction. *Results:* Statistically significant correlations among expertise and inspection outcomes were found in several cases. *Conclusion:* The effect of expertise is observable in the inspection outcome and thus can be used in for software quality management purposes.

1 Introduction

Software development team members' expertise is an essential factor to the success of a software project. Skilled developers are likely to produce better software than less skilled ones. Good code reviewers are likely to detect more defects than bad ones. In this paper, we assess the impact of practitioners' skills in the context of code reviews performed on component-based (CB) software. The reported results are part of a wider experiment, briefly described in the following section, where we assess practitioner's performances in the development, quality control and integration of software components, and compare them with an independent assessment of their skills.

A code review is a peer review of source code intended to detect defects before the testing phase begins, thus improving overall code quality. There are a number of code review processes being used in industry. Fagan inspections [1, 2] are considered seminal in this area. Other processes have emerged since then, that try to lower the costs involved in code inspections without sacrificing their benefits. Examples include conducting inspections offline, thus skipping the inspection meeting [3], or performing phased inspections, where the inspectors focus on a specific class of defects [4], although the latter technique has been criticized for being more costly than conventional inspections [5].

Understanding what drives inspections' success has been a long time concern in the software community. Based on data collected from over 6000 inspections, Weller

studied the impact of the inspection process on software quality [6]. Among several other remarks, he pointed to the familiarity of the inspection team with the artifact being inspected as a key factor in inspection success. We may regard this as kind of domain expertise. Siy observed that while structural changes were largely ineffective in improving the results of inspections, the inputs for those inspections (the reviewers and code being inspected) were far more influential in the inspection outcome [7]. These findings were further explored in [8], to conclude that better inspection techniques, rather than processes, were the key to improving inspection effectiveness. Biffel and Halling combined reviewers' expertise measures (software development skills, experience and an inspection capability pre-test) with different code inspection techniques [9]. While they could not find significant relationships between development skills and experience and inspectors performance, they found the inspection capability pre-test useful to optimize the inspection outcome by selecting ideal inspection teams. They also identified performance differences related to alternative code reading techniques, a result that is consistent with the findings of Laitenberger and DeBaud, in their systematic review on code inspections reading techniques [10]. Sauer et al. identified individual's task expertise as the primary driver of review performance [11].

In a totally different context (social psychology), Kruger and Dunning observed that the skill of a person in performing a task is closely related to the required skill to assess his own performance in the same task [12]. If we instantiate this insight into code production and code reviewing, we would expect the best programmers to also be the most effective code reviewers.

2 Problem statement

Our global goal is to analyze the outcome of a CB development process, for evaluation purposes, with respect to the impact of practitioner's expertise on defect introduction and detection, from the point of view of a project manager (in this case, the research team), in the context of an academic simulation of a component marketplace.

In this paper we are concerned with the impact of practitioners' expertise in the outcome of CB software code reviews performed at the component level. We are seeking evidence on possible causal relationships between the expertise of practitioners involved in the code inspections and the diversity of defects reported during those inspections (**Fig. 1**). All inspections were carried out by a review team (RT) which included the development team (DT) and a peer team (PT). PTs consisted of developers of a different component, participating as independent code reviewers. The remaining process and product inputs are fairly similar for all code inspections, to minimize possible confounding effects.

We consider four potential causal relationships. The expertise of the development team (1) may have a negative effect on the diversity of defects found. The rationale is that expert developers tend to introduce fewer defects on their code. Conversely, peer reviewers expertise may have a positive effect on the defects diversity (2). A similar rationale leads to the possible causal effect between the expertise of the review teams

as a whole (3) and defect diversity. Finally, we consider the difference of expertise between the developer team and the peer team (4) as a negative effect on defect diversity. If the expertise of the developers is higher than that of their peers, defect diversity is expected to be smaller than when the opposite occurs.

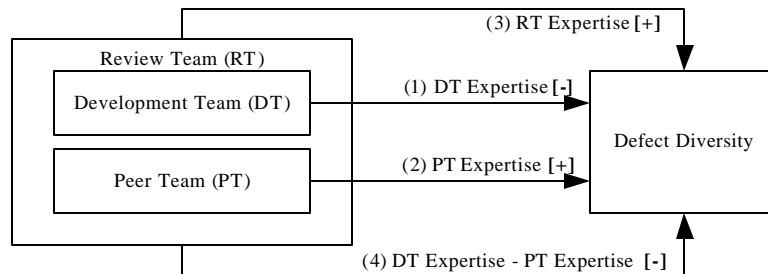


Fig. 1. Exploring the impact of practitioner’s expertise in the outcome of a review process.

3 Experiment planning

3.1 Context selection

This experiment occurred in the context of a Software Engineering course held at the Universidade Nova de Lisboa, during the Spring semester of 2005. This course is offered on the 8th semester of their 5-years informatics degree. According to the new harmonized academic curricula adopted in Europe (Bologna model), these are 2nd cycle degree (MSc) graduate students. The course’s project consisted in developing a CB elevator system simulator from requirements definition to final product delivery. The programming language was Java, well-known to all subjects in the experiment.

Among other activities, the development process included a Fagan inspection performed on all the developed components. While PT members were knowledgeable in the inspected code basic requirements, they were not developing an alternative implementation of that same component, to avoid biasing their review, besides better reproducing industry practice.

Standard Fagan inspection roles were assigned to the four RT members. The DT members got the moderator and author roles and the PT members the remaining ones (reader and recorder). An extensive checklist of common defects in Java programs was distributed (and its contents explained) to all RTs before code inspections took place. This paper focuses on the analysis of the outcome of these inspections.

3.2 Hypothesis formulation

The observations on the problem statement section lead us to testing four different basic hypotheses, to assess the effect of practitioners’ expertise on the outcome of the code inspection, in terms of the inspected defects diversity. We identify the hypothe-

ses as HA, HB, HC, and HD. For each of them, we formulate both a null and an alternative hypothesis (e.g. HA₀ and HA₁).

As we shall see in the next section, we will break down each of these hypotheses into several specialized versions, to try out different expertise assessment metrics.

HA ₀ :	Developer skill has no effect on the inspected defect diversity.
HA ₁ :	Developer skill has an effect on the inspected defect diversity.
HB ₀ :	Peer skill has no effect on the inspected defect diversity.
HB ₁ :	Peer skill has an effect on the inspected defect diversity.
HC ₀ :	Reviewer expertise has no effect on the inspected defect diversity.
HC ₁ :	Reviewer expertise has an effect on the inspected defect diversity.
HD ₀ :	The gap of expertise between developer and peer has no effect on the inspected defect diversity.
HD ₁ :	The gap of expertise between developer and peer has an effect on the inspected defect diversity.

3.3 Variables selection

Independent variables. The basic independent variable of this experiment is the subjects' expertise. We use two measures of our subject's expertise: their *Average Grade (AG)* throughout their academic path, based on the independent evaluation our subjects received in over 30 different courses, and the *number of semesters (NSem)* it took them to complete those courses. We assume that there is a higher merit in obtaining a given *AG* in the *recommended number of semesters (RSem)*, than in a higher *NSem*. The *Simple Weighted Average Grade (SWAG)* and the *Complex Weighted Average Grade (CWAG)* expertise metrics, defined below, follow this rationale. Note that *SWAG* causes a bigger penalty than *CWAG*, as *NSem* increases.

$$SWAG = AG \times \frac{RSem}{Max(RSem, NSem)} \quad CWAG = AG \times \sqrt{\frac{RSem}{Max(RSem, NSem)}}$$

For each of the RTs, concerning expertise, we use the best subject, the worst subject, and the average within the team. Finally, we also consider the difference between the expertise of the DT and that of the PT as an independent variable. In summary, we have 3 alternative rating schemes for grades, and 3 ways of combining grades within teams. This implies that we have 9 different ways for quantifying our independent variable (the expertise). These alternatives are used for each hypothesis under test.

Dependent variables. The dependent variables used in this experiment represent the diversity of defects found during code inspection. A defect classification checklist was distributed to all participants. The checklist contained 16 different defect classes, which were then subdivided into a total of 81 different defect codes. In summary, our dependent variables are:

- *NDDClass*: the number of different defect classes reported in the inspection;
- *NDDCode*: the number of different defect codes reported in the inspection.

At first, *NDDClass* may seem unnecessary, given the usage of a finer grained measure (*NDDCode*). However, if two code inspections report a similar number of different defect codes, but one of them uses a lot less defect classes than the other, it may be the case that this reflects a lower coverage of the kinds of problems to be found during the inspection. We used *NDDClass* to detect this kind of problem, should it occur.

3.4 Selection of subjects

The 87 subjects participating in this experiment are a convenient, but also representative sample of the informatics students which annually graduate from our university (the *numerus clausus* for the informatics degree is 160, and the number of students graduating each year is around 60).

3.5 Experimental design

Regarding the experiment instrumentation, the calculation of subjects' expertise was done upon the data available from the university's academic database. The information concerning code inspections was collected from the normalized inspection reports submitted by subjects after they performed the Fagan inspections. Potential threats to validity [13], and how they were dealt with, are identified throughout the paper.

4 Data Collection

Preparation. The subjects were not aware of the aspects being researched, at the time they participated in the experiment, as this could jeopardize the validity of the results. They were only aware of our intention to use the data collected during the project.

Prior to the implementation of the components that were later inspected, subjects received a Java coding style guide, along with the set of standard public APIs for the components (specified as Java interfaces). Concerning code inspections, besides the referred checklist, subjects received a report template, so that they would perform the inspection and write down the report in a standardized fashion. They also received training on how to perform Fagan inspections, prior to actually starting them.

Execution. The experimental process was not allowed to disturb in any way the subjects' activities in the projects. Subjects performed their normal tasks while developing this project, from requirements specification down to project delivery. Code inspection data was collected from the project's deliverables, which was checked-in in a contents management system made available to students.

Data validation. In the beginning of the semester, there were 93 students enrolled in the course. Five of them dropped out before the experiment started, and one also gave up before turning in the first implementation of his group's component. The remaining 87 students completed the project and are the subjects of this experiment. They were paired into 44 DTs. 43 of those DTs produced components that were inspected. The deliverables of these 43 inspections were used to collect the dependent variables.

5 Data analysis

We summarize the most relevant findings of our hypotheses tests. Further details are available at <http://ctp.di.fct.unl.pt/QUASAR/Projects/CBSE/CodeInspections/>.

5.1 Data set reduction

Outlier and extreme values can change our view on the relations between dependent and independent variables. For each dependent variable, we conducted a linear regression analysis using the average, best and worst cases of the independent variables. We repeated this analysis for each of our hypotheses and flagged as outliers those cases where the standard residual is greater than 1.5 times the standard deviation. This resulted in the removal of four cases in our analysis, with each of the dependent variables. The outlier removal process is illustrated in **Fig. 2**, where cases 15, 19, 25, and 38 are removed.

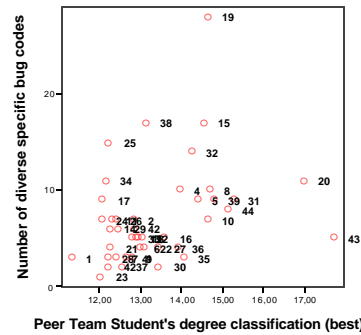


Fig. 2. Number of diverse specific bug codes, by PT expertise, including outliers.

5.2 Normality tests

We used the Kolmogorov-Smirnov (with the Lilliefors correction) to check normality. Using a confidence interval of 99% (test significance = 0.01), we can not reject the normality hypothesis, both for the independent and dependent variables. Therefore, we can use parametric tests, such as the Pearson correlation coefficient.

5.3 Hypothesis testing

We started by performing correlation analysis, using the Pearson coefficient, among each of the independent and the dependent variables, to determine whether or not a relationship exists. These correlations were not significant with the predictors of hypotheses HA and HC. As such, neither the influence of the DT skill, nor the influence of the overall RT skill in the outcome of the reviews, in terms of the diversity of the defect codes and classes, was confirmed.

From **Table 1** we can observe significant correlations between our independent and dependent variables for hypotheses HB and HD. Most of the candidate predictors for HB have a significant positive correlation of above 40%. This relationship is observed both with *NDDCode* and *NDDClass*. The predictors for HD have a significant negative correlation with the dependent variables. These correlations are stronger with *SWAG* and *CWAG* than with *AG*. Again, the same effect is observable both with *NDDCode* and *NDDClass*.

Table 1. Pearson correlations for the variables in hypotheses HB and HD, considering 39 cases (the outlier values referred in section 5.1 were removed, before the correlation analysis).

		HB						HD					
		PT						Diff_DT_PT					
		AG	sig.	SWAG	sig.	CWAG	sig.	AG	sig.	SWAG	sig.	CWAG	sig.
NDDCode	Avg.	.469	.003	.429	.006	.462	.003	-.402	.011	-.454	.004	-.471	.002
	Best	.419	.008	.385	.016	.413	.009	-.409	.010	-.393	.013	-.419	.008
	Worst	.479	.002	.441	.005	.470	.003	-.350	.029	-.470	.003	-.468	.003
NDDClass	Avg.	.416	.009	.378	.018	.407	.010	-.356	.026	-.433	.006	-.443	.005
	Best	.394	.013	.315	.051	.353	.028	-.376	.018	-.370	.020	-.395	.013
	Worst	.392	.013	.413	.009	.430	.006	-.293	.070	-.451	.004	-.451	.004

We further explored the HB and HD hypotheses, to check for significant differences observed in different groups of code reviews, using the ANOVA test. We started by computing the quartile values for the independent variables, and assigned the reviews to the respective quartile group. For each test, we had four groups with a growing expertise of the PT (in hypothesis HB), or with a growing difference between the expertise of the DT and the expertise of the PT (in hypothesis HD). The latter ranges from a negative value (DT has less expertise than PT) to a positive one (DT has a higher expertise than PT). **Table 2** shows an example of this means comparison test, for hypothesis HD.

Concerning HB, we observed a variation among the different review groups that always followed the same pattern. The reviews on the 4th quartile (the ones with the most expert peer teams) were always the ones with the highest *NDDCode* and *NDDClass*. Except when using predictors based on the worse PT element, or the average value of *SWAG*, these differences were statistically significant. *NDDCode* and *NDDClass* showed an increase ranging from 36% to 111%. This trend is not visible in the first three quartiles, for some of the used metrics. The scatterplot presented on **Fig. 2** is an example of a typical distribution of defect diversity vs. PT expertise. In summary, we can reject the null hypothesis HB_0 . We were able to find several measures of the expertise within the PT which can be used as predictors of the diversity of the reported defects.

With respect to HD, we observe the opposite pattern. With the expertise functions being used, the average number of different reported bug codes and classes decreased between 19% and 49%, when comparing the first with the last quartiles. In other words, the number of diverse defect codes and classes decreases as we move from DTs with lower expertise than their PTs to the opposite case. As such, we can reject the null hypothesis HD_0 . We found several measures of the difference between the expertise of the members of DT and PT which can be used as predictors of the diversity of the reported defects.

Table 2. Mean number of diverse defect codes found during code inspections. The difference between average AG of the DTs and PTs metric is used to place the PTs into the respective quartiles. Note that on the 1st quartile, DT has a much lower expertise than PT, while on the 4th, PT has a much lower expertise than DT.

Quartile	Mean Diverse Defects	N	Std. Dev.
1st	7.00	10	2.828
2nd	5.40	10	3.688
3rd	6.10	10	2.558
4th	4.78	9	2.819
Total	5.85	39	3.005

6 Discussion

HA. We expected the best developers to produce components with fewer defects, but this was not confirmed. This result may be explainable in different ways. We did not use any information concerning neither the relative severity of the defects found in this analysis, nor their expected impact on maintenance. Moreover, we used defect code and class diversity, but not the actual number of reported defects in this analysis. Therefore, it may be the case that our dependent variable is too simplistic. It may also be the case that, because PTs were also part of the RTs, their expertise countered the effect of a lower variety of problems with that of a higher efficiency in finding them. A way to circumvent this would be to have several inspections being performed on the same artifacts by different teams, but this was not feasible in our context.

HB. As expected, we observed that the expertise of the PT does have a positive effect on the variety of problems uncovered during code inspections. We also note that the average and higher element expertise within the PT have stronger correlations with the outcome of the review than the expertise of the “weaker” element of the PT. Along with the significant boost of results with the PTs on the best quartile, this increases our confidence on the positive effect of expert peer reviewers in the reviewer team and also points to a small effect of “leadership” within those teams.

HC. The expertise of the whole review team did not show a significant relationship with the outcome of the review. The considerations concerning a possible oversimplification of our dependent variable, combined with the cancellation effect also described with respect to HA may be responsible for this discrepancy between the expected result and the outcome of this experiment.

HD. As expected, when PTs of low expertise analyze the work of DTs with a higher expertise, the outcome of the code review shows a lower variety of defects found. Conversely, more defects are found in inspections where the PTs have a higher expertise than the one of the DTs. A potential leadership effect of a reviewer over the others is not visible from the data analyzed while testing this hypothesis.

With the experiment design of this last hypothesis, we have an alternative perspective on the inspection group dynamics, when compared to hypothesis HC. On HC we had no indication of how the expertise was distributed within the group, thus being

vulnerable to the cancellation effect occurring when (i) having good experts examining their own code and not finding many problems with it, because they were not there, or (ii) weaker programmers examining their own code and not realizing the problems in it. Both situations lead to a cancellation effect that might explain the unexpected results with hypothesis HC.

There is a curious effect in the evolution of the variety of defects found between the second and third quartiles of HD (the second quartile has DTs with a lower expertise than their PTs, while the third inverts this relationship). One could expect the variety of defects to be lower on the third quartile, following the tendency found from the first to the fourth quartiles. However, the expertise level is very close, within groups 2 and 3. Therefore, it may be the case that it is the domain level expertise that dominates the outcome of the inspection. With a better knowledge of the deliverables being inspected, allied with a slightly better expertise than their peers, the authors may be responsible for this locally increased benefit of the code review. As the gap of expertise between DT and PT members widens, this effect would be mitigated by the dominating effect of the higher code quality and lower external reviewer expertise.

7 Conclusions

We described an experiment carried out to help understanding the effect of practitioner's expertise in the deliverables produced in the context of CB development.

We focused our attention on the outcome of code inspections, and, in particular, on the variety of problems reported during those inspections. We confirmed the expected positive effect of the expertise of the peer review teams in the outcome of the reviews, observable through the increased variety of defects found when peer experts were available. We also confirmed that having expert peers collaborating in the inspection of components developed by less skilled peers has a positive impact on the outcome of the review. Moreover, there is also a learning effect, not studied here but vastly commented on the literature, when combining experts with non-experts. This is also expected for the opposite case, where non-experts participate on the review of code developed by experts. However, in this case, a lower variety of defects is found, both because the code is likely to have a higher quality, and because the external reviewers have less capacity to detect its problems. Given the main goal of inspections (maximizing defect detection), the results are poorer.

When observed in isolation, the expertise of the DTs did not show a significant relationship with the variety of problems found. The expertise of the RTs was also not shown to be a good indicator of the outcome of the inspection. Further research is required to determine whether these were the results of cancellation effects of expertise, or if more sophisticated review outcome metrics should have been used here.

As future work, we expect to expand on this experiment by exploring this interpretation of why two of our hypotheses were not confirmed. The deliverables of the project that served as a basis for this experiment include some details that were not explored in this paper, such as code complexity metrics, and the practitioners' assessment of the potential impact of the problems reported. We plan to further explore these data to

strengthen the conclusions reported here and to explore other related hypotheses on the effect of expertise throughout the development process.

Acknowledgments

This work is sponsored by the FCT STACOS project (POSI/CHS/48875/2002).

References

1. Fagan, M.E., *Design and Code Inspections to Reduce Errors in Program Development*. IBM Systems Journal, 1976. 15(3): p. 182-211.
2. Fagan, M.E., *Advances in Software Inspections*. IEEE Transactions on Software Engineering, 1986. 12(7): p. 744-753.
3. Parnas, D.L. and Weiss, D.M., *Active Design Reviews: Principles and Practices*. Journal of Systems and Software, 1987. 4(7): p. 259-265.
4. Knight, J.C. and Myers, E.A., *An Improved Inspection Technique*. Communications of the ACM, 1993. 36(11): p. 51-61.
5. Porter, A. and Votta, L., *What Makes Inspections Work?* IEEE Software, 1997. 14(6): p. 99-102.
6. Weller, E.F., *Lessons from Three Years of Inspection Data*. IEEE Software, 1993. 10(5): p. 38-45.
7. Siy, H., *Identifying the Mechanisms to Improve Code Inspection Costs and Benefits*, PhD Thesis, University of Maryland, USA. 1996.
8. Porter, A., Siy, H., Mockus, A., and Votta, L., *Understanding the sources of variation in software inspections*. ACM Transactions on Software Engineering and Methodology, 1998. 7(1): p. 41-79.
9. Biffi, S. and Halling, M. *Investigating the Influence of Inspector Capability Factors with Four Inspection Techniques on Inspection Performance*. in *Eighth IEEE International Symposium on Software Metrics (Metrics'02)*. 2002.
10. Laitenberger, O. and DeBaud, J.-M., *An Encompassing Life-Cycle Centric Survey on Software Inspection*. Journal for Systems and Software, 2000. 50(1): p. 5-31.
11. Sauer, C., Jeffery, R., Land, L., and Yetton, P., *The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research*. IEEE Transactions on Software Engineering, 2000. 26(1): p. 1-14.
12. Kruger, J. and Dunning, D., *Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments*. Journal of Personality and Social Psychology, 1999. 77(6): p. 1121-1134.
13. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A., *Experimentation in Software Engineering: An Introduction*. Vol. 6. 1999, Boston, EUA: Kluwer Academic Publishers. 224 pages.