Efficiency of Review Meetings

Haris Causegic, Aretina Iazzolino, Andreas Korge, Josip Ledic and Thommy Zelenik
University of Stuttgart
Institute of Software Technology
Stuttgart, Germany
Email: inf82831@stud.uni-stuttgart.de

***Abstract*—The abstract goes here.**

I. Introduction

Traditional code and document inspection in terms of discovery and comprehension of mistakes and bugs is one of the most important processes in software development. In addition to the intuitive approach of manually discovering mistakes, side effects and bugs in software and documents, there are various techniques to further enhance code and documentation quality, ranging from spell-checking, static and dynamic code analyses, automatic code quality monitoring to project check-in policies. These techniques have one thing in common. They can mostly be applied (semi-)automatically without the need of interacting with other fellow developers.

Another technique for improving code quality, that utilizes the *many eyes see better than two* principle are review meetings. These developer meetings generally take place after each developer independently reviewed the code or document in question. The idea of these meetings is that all independently discovered issues get accumulated, presented and discussed in a public manner. Earlier work on code review meetings has shown that such review meetings are useful for discovering potential bugs, improving code quality and enforcing standards (software reviews: the state of the practice). The question that yet remains to be answered is if code review meetings are an efficient way of improving code quality in general.

II. Related Work

In this section we discuss the related work with respect to the efficiency of technical review meetings and the relevance of the meeting sessions. Prior work has qualitatively analyzed the technical review process used by large software systems.

A. Review without meeting session

Without a review meeting the reviewers would just send their list of findings via tools developed for supporting code review to the author of the artifact or choosing one reviewer taking care of all the review work. The following papers show the influence on the quality of the artifact when using this method.

Baker discussed a procedure for reviewing code changes that are made to a software product as it moves through its life cycle. Facing a decline of customer confidence in his product his team came up with the idea of using a revision control system to identify all changes made to the product and a line by line review of the changes by the project manager. He noted that this methodology came with several advantages one of them being the efficiency with which the quality improvement was made. A single reviewer becomes very efficient in his reviews due to practice at reviewing code and to a thorough understanding of the product. The fact they used the product manager as the single reviewer facilitated him getting an insight into the whole project encompassing each engineer's performance, the complexity of the tasks, progress of the project and also bettering his scheduling skills. The employees liked this methodology because of the manager getting an understanding of the effort in their work, capabilities and identifying areas that needed improvement.

By using this methodology a 40% reduction in the number of bugs at a cost of 1-2% of the project was achieved [**?**].

Mcintosh et al. studied the relationship between software quality and:

- the proportion of changes that have been code reviewed (code review coverage), and
- the degree of reviewer involvement in the code review process (code review participation)

through a case study of the large Qt, VTK, and ITK open source systems. The Gerrit-driven code review process for git-based software projects has been used which is tightly integrated with test automation and code integration tools. They extracted code review data from the Gerrit review databases of the studied systems and linked the review data to the integrated patches recorded in the corresponding version control systems. They built Multiple Linear Regression (MLR) models to explain the incidence of post-release defects detected in the components of the studied systems having the goal to understand the relationship between the explanatory variables (code review coverage and participation) and the dependent variable (post-release defect counts).

They found out that low code review coverage and participation are estimated to produce components with up to two and five additional post-release defects respectively [**?**].

Bacchelli et al. empirically explored the motivations, challenges, and outcomes of tool-based code reviews. They observed, interviewed, and surveyed developers and managers and manually classified hundreds of review comments across diverse teams at Microsoft using discussions within the review tool CodeFlow. They found out that the main motivation

for reviews despite finding defects are knowledge transfer, increasing team awareness, and creation of alternative solutions to problems [**?**].

B. Review with meeting session

One common way of quality assurance of documents or code in a software development process is to organize a technical review where several reviewers examine an artifact and discuss the findings together in a two to three hour review meeting. Although it has been shown that a technical review can find bugs and issues efficiently it is unclear how relevant the meeting session is where reviewers discuss their findings.

III. Experimental Design

A. Research Questions

First, we wanted to see, whether review meetings help the software development process in terms of quality management.

RQ1: Are review meetings effective in regards to improving document or code quality?

Next, we were interested in how efficient these review meetings are in regards to time and monetary requirements.

RQ2: How efficient are code review meetings as a means to increasing document or code quality?

Additionally to direct improvements to the code via the review reports, other studies concerning the topic suggest, that the execution of reviews may provide additional benefits such as knowledge transfer, increased team awareness, or creation of alternate solutions [**?**].

RQ3: Which positive effects besides improving code quality do review meetings entail?

B. Hypotheses

Initially, we defined three hypotheses and their according null hypotheses. The first hypothesis refers to RQ1. We wanted to understand if review meetings help improving document quality at all and thus only look at the effectiveness.

H1: Review meetings are effective in improving document quality.

The respective null hypothesis assumes that there is no significant improvement in document quality.

H10: Review meetings are not effective in improving document quality.

Next we were interested in whether or not review meetings are also efficient in improving document quality, since a group of usually five or six software developers is required and thus entails high monetary costs.

H2: Review meetings are an efficient way of increasing document quality.

The null hypothesis for H2 presumes that review meetings are not efficient in improving document quality.

H20: Review meetings are not an efficient way of increasing document quality.

Finally, we defined our third hypothesis concerning the research question of whether or not review meetings bring positive side effects to the table.

Subjects	No Review Meeting	Review Meeting
”SoPra” Group 1	x	x
”SoPra” Group 2	x	x
...	x	x

TABLE I
THE ONE FACTOR TWO TREATMENT DESIGN USED FOR OUR EXPERIMENT.

H3: Review meetings have a positive impact besides the direct improvement to document quality.

The corresponding null hypothesis, assuming that there are no positive side effects, looks as follows:

H30: Review meetings do not have any positive impact besides the direct improvement to document quality.

C. Design

Quantitative and qualitative analysis.? One factor 2 treatment.

For our quantitative analysis we used a one factor two treatment design. The first treatment is the control treatment of dong no review meetings, while the second treatment is the execution of a review meeting. Since doing no review does not influence the subjects, each group was assigned to both treatments.

D. Objects

The idea of our experiment was that we analyse the review report that result out of the review meetings and compare them to the merged lists of findings from each individual reviewer.

1) Review Meetings: The review meetings are conducted by students of the University of Stuttgart within the scope of the ”Software Praktikum”. Each review group consists of five people, being three reviewers, one moderator and one scribe, who is also representing the authors. Before the review meeting itself, all reviewers have to inspect the document and create a list of findings using a review tool [**?**] tool. The reviews are expected to have a duration of TODO:90 minutes. During the reviews, the scribe also uses a review meeting tool to gather the findings and in the end creates a final review report.

2) Review Tools: We looked into the following review tools: *RevAger: Collaborator* TODO Thommy: Find more review tools, explain them, decide for RevAger (and find a reason).

E. Experiment Procedure

This part describes the procedure of the experiment including the prerequisites and assumptions that made this succession of events reasonable and adequate. The experiment consists of multiple phases to get both quantitative and qualitative data as a result.

1) Prerequisite: The ”Software Praktikum” teams were formed before the actual experiment and became familiar with the theory of review meetings and the review tool *RevAger* beforehand. They already participated in a review meeting during an earlier phase of their projects, therefore it is reasonable to assume that they know how to do a review meeting properly.

2) Preparation phase: During the preparation phase each team worked on their project independently from each other and unknowingly of our experiment. They each created a specification document as a result of their requirements analysis. Because they technically all had to implement the same piece of software, we were able assume that they had a good level of understanding of the underlying problem and the requirements for this piece of software. At this point, the "Software Praktikum" teams normally would've gotten each other's specification documents to review in a *round-robin* manner.

To create a homogeneous setting for our experiment, we gave our own specification document to all teams instead. This specification document was created by the authors of this paper independently from the experiment participants, and it is safe to assume that it contains a multitude of errors and mistakes due to it being the very first iteration of a specification document and due to the authors not being fault-free either. Hence we could do a more realistic evaluation, because all teams reviewed the same document.

3) Active phase: During the active phase of the experiment the actual review meetings took place, which we observed silently. The participants each presented their findings, which they had worked out singlehandedly before the meeting, in front of their teams. The resulting quantitative and qualitative data was used to determine answers for the underlying research questions, as described in the following sections.

F. Data collection Procedure

We classify our data collection in two groups, quantitative and qualitative data collection. The quantitative data for the control treatment is gathered by taking the findings lists of each reviewer in a review group and merging these lists together, eliminating duplicates. On the other hand, the quantitative data for the review treatment equals the findings list that results out of the review. Additionally, the RevAger tool automatically measures the elapsed time. The tool also saves the participants and their respective roles.

The structure of a finding consists of a description, an aspect (what reviewers should have a special focus on, e.g. completeness), a reference (where to find it in the document) and an importance rating.

For the qualitative data collection we conduct interviews with all students participating in the review meeting. The interviews are all conducted immediately after every review meeting, in which every participant is interviewed individually. Before every interview, the participants have to sign a consent form. All audio during the discussion is recorded if agreed by participants and transcribed verbatim for the analysis. Audio records are stored securely and are not intended to be published within this work. Additionally, every participant has to fill a demographic questionnaire, which allows us to gather information about participants semester, education degree, previous review knowledge and review experience etc.. For our interview we prepared various quesitons which relate to the following subject areas: *theoretical review knowledge*, *practical review experience*, *subjective view of attended review meeting*, *subjective view of benefits and drawbacks of attended review meeting*.
The interview question model that we use is the funnel model: Opening the interview with open questions and moving towards more detailed and specific questions. Through this interview design, participants are faced with increasingly difficult questions. Using this approach ensures to find out more details about the participants attended review meeting. /TODO: interview structure, grounded theory

G. Analysis Procedure

For the analysis of the findings, we decided to evaluate them manually, due to the fact that there is no objective way of rating the quality of a finding. The evaluation will be executed by three of the study conductors independently and the final result is determined by establishing the median of these.

In order to rate the individual findings, we constructed the following classification scale, which can also be seen in table III: The weighing of the findings are worth 1 for good, 2 for besides error, 3 for main error and 5 for crucial error. If a rating is incorrect, we will instead assign the value for the correct weighting and decrease it by 1. If a finding itself is incorrect (e.g. claims to have found a mistake, where there is none), the finding will recieve a rating of -1.

Next, we tried to find formula that determines the overall quality of a finding list. It is important to consider both the amount of findings, but also the overall quality of these findings. Altough more findings generally are a positive result, we decided that the overall quality is even more important (e.g. to reduce stacking of rather unimportant "good" findings). Therefore our focus is set more on the overall finding quality, meaning that we rate critical and main errors higher. We defined the amount of findings a_L by:

$$a_L = \sum_L$$

Where L is the merged list of all findings, without duplicates.

For the overall quality q_L we determined the arithmetic mean of the accumulated weightings of the findings.

$$q_L = \frac{1}{a_L} \sum_{f \in L} w_f$$

Where w_f is the weighting (see table III) of finding f.

In order to evaluate the difference between two finding lists, we defined the following function:

$$f(L_1, L_2) = \frac{a_{L_2} - a_{L_1}}{a_{L_2} + a_{L_1}} + (q_{L_2} - q_{L_1})$$

While in theory $f \in [-7, 7]$ holds with $\frac{a_{L_2} - a_{L_1}}{a_{L_2}} \in [-1, 1]$ and $q_{L_2} - q_{L_1} \in [-6, 6]$, in reality the values we will recieve will be very small. Suppose the reviewers assemble 100 unique findings beforehand and the review meeting results in 5 new findings (which would be a unusually high amount), our first term would have a value of 0.0244. As for the second term, suppose the average rating for the findings was 3.0 and 10 classification errors occured, we would receive a final quality rating q_{L_1} of 2.951. Assuming that the review

Classification	Rating
'Good'	1
'Besides Error'	2
'Main Error'	3
'Critical Error'	5
'Finding Error'	-1
'Classification Error Penalty'	-1

TABLE II
RATING TABLE FOR THE EVALUATION OF FINDINGS.

meeting corrects all these classification errors, the second term would add up to 0.049. So overall our improvement is 0.0734. Assuming these values are realistic, our previously mentioned focus on the quality would be given with approximately 2:1. Considering these values, we now can define our hypotheses. We define effectiveness as having any improvement from the document without the meeting. We define our H1 Hypothesis on effectiveness as following:

$$\textbf{H1: } f(L_{pre}, L_{post}) > 0$$

Wheres L_pre is the finding list before the review meeting, while L_post describes the finding list that results after the meeting. The corresponding null hypothesis, assuming that there is no improvement resulting out of a review meeting, looks as follows:

$$\textbf{H10: } f(L_{pre}, L_{post}) \leq 0$$

As for efficiency, we first have to consider the time and monetary costs that result out of a review meeting. Assuming that a review consists of three reviewers, one moderator, one scribe and an author with an average earnings of 100/h per developer, an hour would cost approximately 600, not considering potential costs of the meeting room. Since this research only focuses on the review meetings themselves, we do not consider any costs that arise within a review process outside the meeting itself (e.g. the cost for the reviewers to analyse the specimen). Estimating the cost of an error poses a difficult task. According to [**?**] the cost of an error in the requirements phase increases by 30 to 70 times, when found in the acceptance testing, where a specification error is most likely found in. We assume that correcting an average error in the specification takes about 3 minutes to correct and thus has an initial cost of about 5, which will cost about 250 when found in the acceptance testing. We assume that this is the average cost for a critical error in order to not overestimate the efficiency. Further we assume that a besides error is worth 100 and a main error is worth 150. We define our hypothesis H2, that describes the efficiency of review meetings, as follows:

$$\textbf{H2: } 100 * n < 50 * \sum_{f \in L_{new}} w_f$$

Where n is the amount of developers taking part in the review meeting. L_{new} is the list of newly discovered findings, defined as $L_{new} = L_{post} \setminus L_{pre}$.

H. Validity Procedure

The review teams were assigned semi-randomly out of all "SoPra" participants with only one condition, being that no "SoPra" team members are within the same review group.

In order to reduce subjectivity of the manual evaluation that will take place when weighting the findings, we decided that three study conductors independently rate these weightings. The median of these three ratings will be taken as the final value.

Each group recieved the specification document three days before their respective review meetings. By doing this we strived for having their memories on the specimen as fresh as possible.

IV. Analysis

V. Threats to Validity

A. Conclusion Validity

The threat level to conclusion validity of our experiment is rather high. This is due to the fact that some of our measurements could not be automatically calculated but instead had to be determined manually by the conductors. In order to reduce this threat, we decided to let three conductors determine the values independently and finally took the median as our final result. Also since the specification was created by us, we are biased when evaluating the findings. This threat can be eliminated by separating the persons doing the specification and the persons evaluating the findings. Our calculations on the efficiency hypothesis H2 are mainly based on money estimates, and the costs of developers varies between countries, companies and individual position within a company.

B. Internal Validity

Due to the fact that the participants had to be informed about the experiment to give us their consent, they knew that some part of their work will be evaluated and thus might have been influenced in their behavior by the experiment, e.g. resulting in them making more efforts than they would have done otherwise.

Since we cannot prevent students from communicating with eachother, it is unavoidable that information about the specification or experiment in general could be exchanged.

C. Construct Validity

The construct validity between the treatment and the cause construct is a given, since they are identical, being the review meeting.

As for the validity between the outcome and the construct of the effect, ?

D. External Validity

Our study also contains external threats to validity. Since every group of students in the "SoPra" has to solve the exact same task, all produces code or specifications should ideally be semantically equal and thus our results might not necessarily be generalizable to all kind of software systems.

Another external factor that threatens validity are the participants of the study. These are limited to students of the university Stuttgart and thus represent only a minor part of software engineers.

Group No.	Good Error		Besides Error		Main Error		Critical Error		Finding Error		Classification Error		Time
	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post	Pre	Post	
Group 1	x	y											
Group 2													
Group 3													
Group 4													
Group 5													
Group 6													
Group 7													
Group 8													
Group 9													
Group 10													
Group 11													
Group 12													
Group 13													
Group 14													
Group 15													

TABLE III
RATING TABLE FOR THE EVALUATION OF FINDINGS.

Lastly, the size of the ”SoPra” is very small, since it is only a six month project for three developers and thus is far off from realistic industrial projects.

VI. Results and Analysis

TODO

VII. Conclusion and Future Work

TODO

Acknowledgment

The authors would like to thank...

References

[1] R. A. Baker, Jr., “Code reviews enhance software quality,” in *Proceedings of the 19th International Conference on Software Engineering*, ser. ICSE ’97. New York, NY, USA: ACM, 1997, pp. 570–571. [Online]. Available: http://doi.acm.org/10.1145/253228.253461

[2] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, “The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 192–201. [Online]. Available: http://doi.acm.org/10.1145/2597073.2597076

[3] A. Bacchelli and C. Bird, “Expectations, outcomes, and challenges of modern code review,” in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE ’13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 712–721. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486882

[4] J. M. Stecklein, J. Dabney, B. Dick, B. Haskins, R. Lovell, and G. Moroney, “Error cost escalation through the project life cycle,” 2004.