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 inspection in terms of discovery and comprehension of bugs is one of the most important processes in software development. In addition to the intuitive approach of manually discovering side effects and bugs in software, there are various techniques to further enhance code quality, ranging from static and dynamic code analyses over 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 code review meetings. These developer meetings generally take place after each developer reviewed the code independently. The idea of these meetings is that all independently accumulated results get 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

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: How effective are review meetings 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: Are code review meetings an efficient way of increasing document or code quality?

B. Hypothesis

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

H1: Review meetings are effective in improving document or code quality.

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.

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

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

Next we were interested in whether or not review meetings are also efficient in improving code 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 or code quality.

The null hypothesis for H2 presumes that review meetings are not efficient in imporving code quality.

H20: Review meetings are not an efficient way of increasing document or code 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: Find more review tools, explain them, decide for RevAger (and find a reason).

E. Data collection Procedure

The 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 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.

F. 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 II: 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 II) of finding f.

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

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

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.

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 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:

H1: $f(L_1, L_2) > 0$

G. Validity Procedure

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.

Code quality definition?

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.

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.

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.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.