A Qualitative Exploratory Study of How OSS Developers Define Code Review Quality Technical Report

Oleksii Kononenko* and Olga Baysal†

August 14, 2015

Abstract

In a large, long-lived project, an effective code review process is key to ensuring the long-term quality of the code base. In this work, we study code review practices of a large, open source project, and we investigate how the developers themselves perceive code review quality. We present a qualitative study that summarizes the results from a survey of 88 Mozilla core developers. The results provide developer insights into how they define review quality, what factors contribute to how they evaluate submitted code, and what challenges they face when performing review tasks. We found that the review quality is primarily associated with the thoroughness of the feedback, the reviewer's familiarity with the code, and the perceived quality of the code itself. Also, we found that while different factors are perceived to contribute to the review quality, reviewers often find it difficult to keep their technical skills up-to-date, manage personal priorities, and mitigate context switching.

1 Introduction

This report documents the results of the open coding approach that was a part of the qualitative study organized around an exploratory survey with 88 Mozilla core developers. We present our methodology (Section 2), the results of the card sort (Section 3, Section 4, and Section 5), and the original survey form (attached at the end of this report).

2 Methodology

We conducted an exploratory qualitative study that involved data collection through a survey with professional developers. This section describes the survey design, the participants, and the analysis of the responses in detail.

*School of Computer Science, University of Waterloo, okononen@uwaterloo.ca
†School of Computer Science, Carleton University, olga.baysal@carleton.ca

2.1 Survey Design

The survey consisted of three main parts: nine questions about the developer's demographic background and work practices, three Likert-scale questions related to different aspects of code review, and seven follow-on open-ended questions to allow developers to elaborate on issues raised by the multiple choice questions. Participants were asked to spend 5–10 minutes to complete the survey.

The main goal of conducting the survey was to solicit developer feedback on the perceived quality of code reviews and factors affecting review time, decision, and quality. We also wished to identify key problem areas within the existing review process.

2.2 Participants

We decided to continue our work within the Mozilla project developer community for several reasons: much of our previous work has studied this project and we have good intuition about the system and its development practices, we have made good contacts within the project who are supportive of our research goals, and because Mozilla is a well known, very large, and long-lived open source project.

To identify potential participants for our study, we looked at the 12 month history (from May 10, 2014 to May 10, 2015) of all contributions to the Mozilla project as they are recorded in Bugzilla issue tracking system. Because of the Bugzilla's limitations on the search results, we directly queried Mozilla's Elastic Search cluster that contains the up-to-date copy of Bugzilla data [2]. By processing the queried data, we extracted 3,142 unique email addresses (Bugzilla uses an email address as a unique user identifier). After that, we queried the cluster for each email address to get the information about developer's activity: number of contributions submitted for review and the number of patches that were reviewed by the developer during the studied period. Finally, we used Bugzilla's REST API to extract developers' real names.

We decided to limit our survey to experienced developers who were not new to the project. We computed an experience value as the sum of submitted and reviewed patches. We set a threshold for the experience value at 15 — meaning that anyone with a combined experience of at least 15 patches will pass the filter — which reduced the list of potential participants to 843 (27%) people. To filter out developers who were new to the Mozilla project — regardless of their experience level — we defined familiarity as having contributions (submitted and/or reviewed patches) at least 6 months prior to the beginning of the studied period. This filter further reduced the list of experienced developers to 403 (13%) people.

Once we selected developers whom we wanted to survey, we sent out 403 personalized emails. Each email contained the number of contributions submitted or reviewed during the 12 months period and an invitation to participate in the survey. The survey was open for 3 weeks (from May 29 to June 19, 2015) and received 88 responses (22% response rate).

The beginning of the survey consisted of background-related questions. By analyzing the responses, we found that we had successfully targeted highly experienced developers: about 48% of respondents said that they have more than 10 years of software development experience, while another 26% of them have between 7 and 10 years of experience. Most of the respondents have been performing code review for more than 3 years (67%).

2.3 Survey Data Analysis

We applied a grounded theory methodology to analyze the survey data; as we had no predefined groups or categories, we used an open coding approach. As we analyzed the quotes, themes and categories emerged and evolved during the open coding process.

Author Kononenko created all of the "cards", splitting 88 survey responses into 938 individual quotes; these generally corresponded to individual cohesive statements. In further analysis, authors Kononenko and Baysal acted as coders to group cards into themes, merging themes into categories. For each open-ended question, we proceeded with this analysis in three steps:

1. The two coders independently performed card sorts on the 20% of the cards extracted from the survey responses to identify initial card groups. The coders then met to compare and discuss their identified groups.
2. The two coders performed another independent round, sorting another 20% of the quotes into the groups that were agreed-upon in the previous step. We then calculated and report the coder reliability to ensure the integrity of the card sort. We selected two of the most popular reliability coefficients for nominal data: percent agreement and Cohen's Kappa. Coder reliability is a measure of agreement among multiple coders for how they apply codes to text data. To calculate agreement, we counted the number of cards for each emerged group for both coders and used ReCal2 [1] for calculations. The coders achieved a substantial degree of agreement; on average two coders agreed on the coding of the content in 96% of the time (the average percent agreement varies across the questions and is within the range of 94.2–97.2%; while the average Cohen's Kappa score is 0.68).
3. The rest of the card sort (for each open-ended question) — 60% of the quotes — was performed by both coders together.

3 Results: open ended answers

In Sections 3, 4, and 5, we present the raw results of the open coding approach, including the emerged categories, number of quotes and participants, and some diagrams related to the multiple choice questions, as well as developer demographics. Full results of the data analysis, insights and implications of this work are documented in a paper that is currently under review.

3.1 Question 11: 141 quotes, 54 participants

1. *Code quality: correctness, style, etc.* 69 quotes, 31 participants
2. *Testing:* 19 quotes, 15 participants
3. *Contributor experience:* 3 quotes, 3 participants
4. *Time constraints (shipping deadlines):* 2 quotes, 2 participants
5. *Scope/rationale:* 15 quotes, 12 participants
6. *Conformance to project goals and specifications:* 8 quotes, 4 participants
7. *Reviewer-related: expertise, personality, style* 12 quotes, 7 participants
8. *Discussion:* 2 quotes, 2 participants
9. *Relationship/trust:* 7 quotes, 5 participants
10. *Everything else:* 4 quotes, 4 participants

Average percent agreement: 97.2%

3.2 Question 13: 63 quotes, 43 participants.

1. *Code complexity/quality: correctness, style, API changes, etc.* 24 quotes, 21 participants
2. *Code familiarity:* 4 quotes, 4 participants
3. *Testing:* 4 quotes, 4 participants
4. *Past relationship b/w reviewer and submitter:* 2 quotes, 2 participants
5. *Patch rationale:* 6 quotes, 5 participants
6. *Submitter type:* 2 quotes, 2 participants
7. *Communication b/w reviewer and reviewee:* 1 quote, 1 participant
8. *Bug type:* 6 quotes, 5 participants
9. *Correct reviewer (backlog, personal priorities):* 5 quotes, 5 participants
10. *Tools:* 3 quotes, 2 participants

11 Everything else: 6 quotes, 5 participants

Average percent agreement: 96.6%

3.3 Question 14: 290 quotes, 86 participants.

1. *Code quality: code style, comments, etc.* 90 quotes, 56 participants
2. *Patch complexity:* 30 quotes, 27 participants
3. *Correctness/rationale of the change:* 75 quotes, 50 participants
4. *Testing:* 34 quotes, 31 participants
5. *Performance:* 7 quotes, 7 participants
6. *Integration into code base:* 13 quotes, 13 participants
7. *Architecture/design:* 14 quotes, 13 participants
8. *Security:* 3 quotes, 3 participants
9. *Memory management:* 3 quotes, 2 participants
10. *Usefulness:* 3 quotes, 3 participants

11. *Familiarity with the author:* 4 quotes, 4 participants
12. *Misc:* 14 quotes, 13 participants

3.4 Question 15: 219 quotes, 86 participants.

1. *Human factors:* 37 quotes, 24 participants
2. *Understanding patch/codebase:* 47 quotes, 26 participants
3. *Testing:* 15 quotes, 13 participants
4. *Usefulness:* 3 quotes, 3 participants
5. *Design/architecture:* 13 quotes, 9 participants
6. *Code quality/style:* 19 quotes, 19 participants
7. *Thorough feedback:* 50 quotes, 33 participants
8. *Timeliness:* 18 quotes, 17 participants
9. *Misc:* 9 quotes, 9 participants
10. *Catching bugs:* 8 quotes, 7 participants

3.5 Question 17: 50 quotes, 26 participants.

1. *Tools:* 3 quotes, 3 participants
2. *Human factors:* 7 quotes, 6 participants
3. *Communication:* 4 quotes, 3 participants
4. *Patch rationale:* 2 quotes, 2 participants
5. *Code quality:* 4 quotes, 4 participants
6. *Testing:* 4 quotes, 4 participants
7. *Time constraints:* 7 quotes, 5 participants
8. *Understanding codebase/domain:* 10 quotes, 8 participants
9. *Code complexity:* 4 quotes, 4 participants
10. *Workload:* 2 quotes, 2 participants
11. *Availability of documentation:* 1 quote, 1 participant
12. *Organizational factors:* 2 quotes, 1 participants

3.6 Question 18: 81 quotes, 55 participants.

1. *Time constraints:* 14 quotes, 14 participants
2. *Understanding codebase/domain:* 25 quotes, 21 participants
3. *Human factors:* 9 quotes, 9 participants
4. *Code complexity:* 8 quotes, 8 participants
5. *Tools:* 7 quotes, 5 participants
6. *Patch rationale:* 8 quotes, 8 participants
7. *Workload:* 3 quotes, 3 participants
8. *Context switch:* 5 quotes, 5 participants
9. *Communication:* 1 quote, 1 participant

10. Code quality: 1 quote, 1 participant

3.7 Question 19: 90 quotes, 51 participants.

1. Reviewer recommender system: 1 quote, 1 participant
2. Better "diff" tools: 13 quotes, 11 participants
3. Automated code style checker: 21 quotes, 15 participants
4. Warning system of error-prone code locations: 1 quote, 1 participant
5. Debugging/testing; 5 quotes, 4 participants
6. Online review tool: 4 quotes, 4 participants
7. Better dev environments: 18 quotes, 12 participants
8. Automated build system: 2 quotes, 2 participants
9. Integrated code navigation: 8 quotes, 6 participants
10. Better review tools: 5 quotes, 5 participants
11. Patch history: 2 quotes, 2 participants
12. Junk: 10 quotes, 8 participants

4 Results: multiple choice answers (Likert scale)

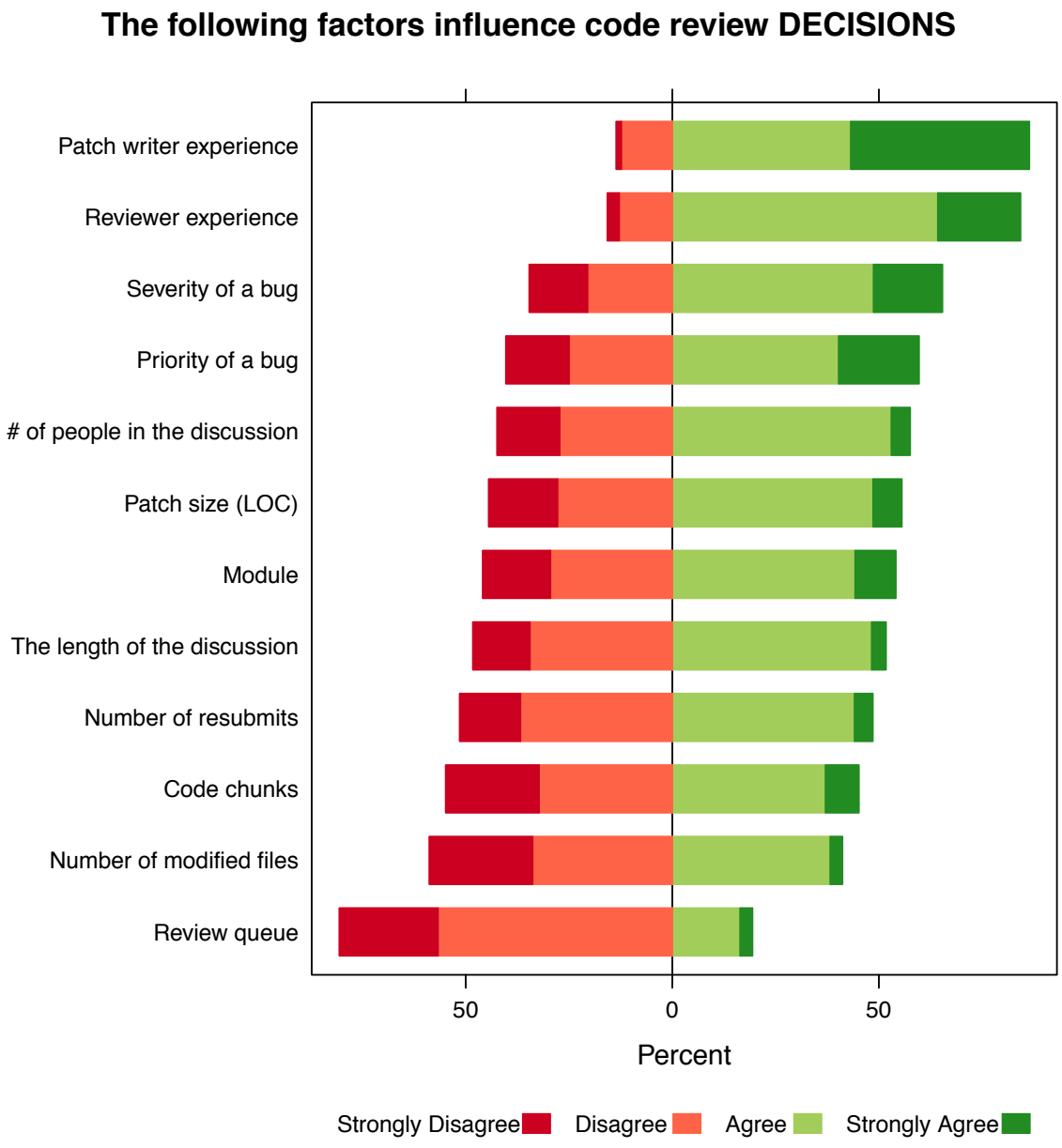

Figure 1: Q10: Factors influencing code review decisions.

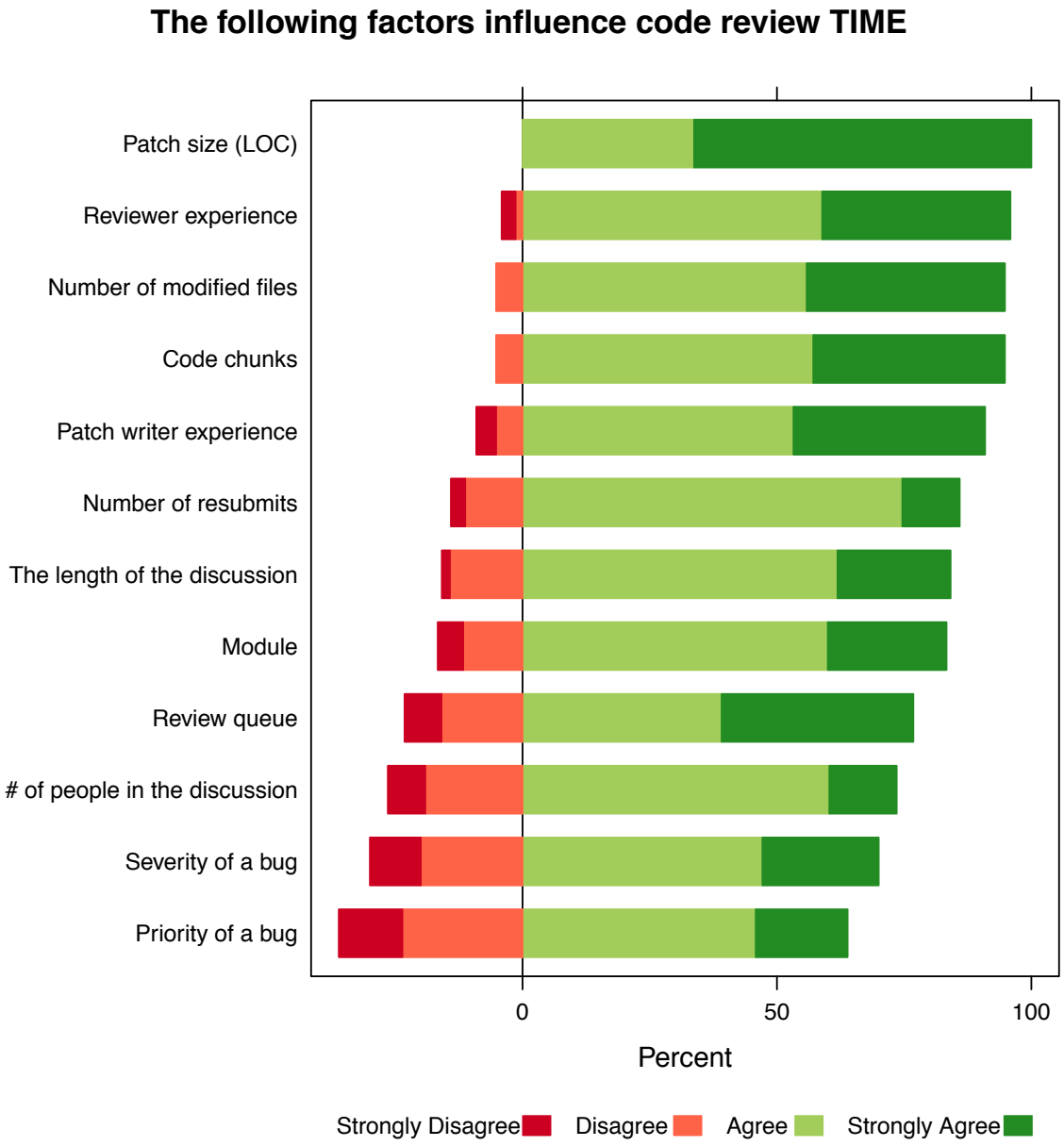

Figure 2: Q12: Factors influencing code review time.

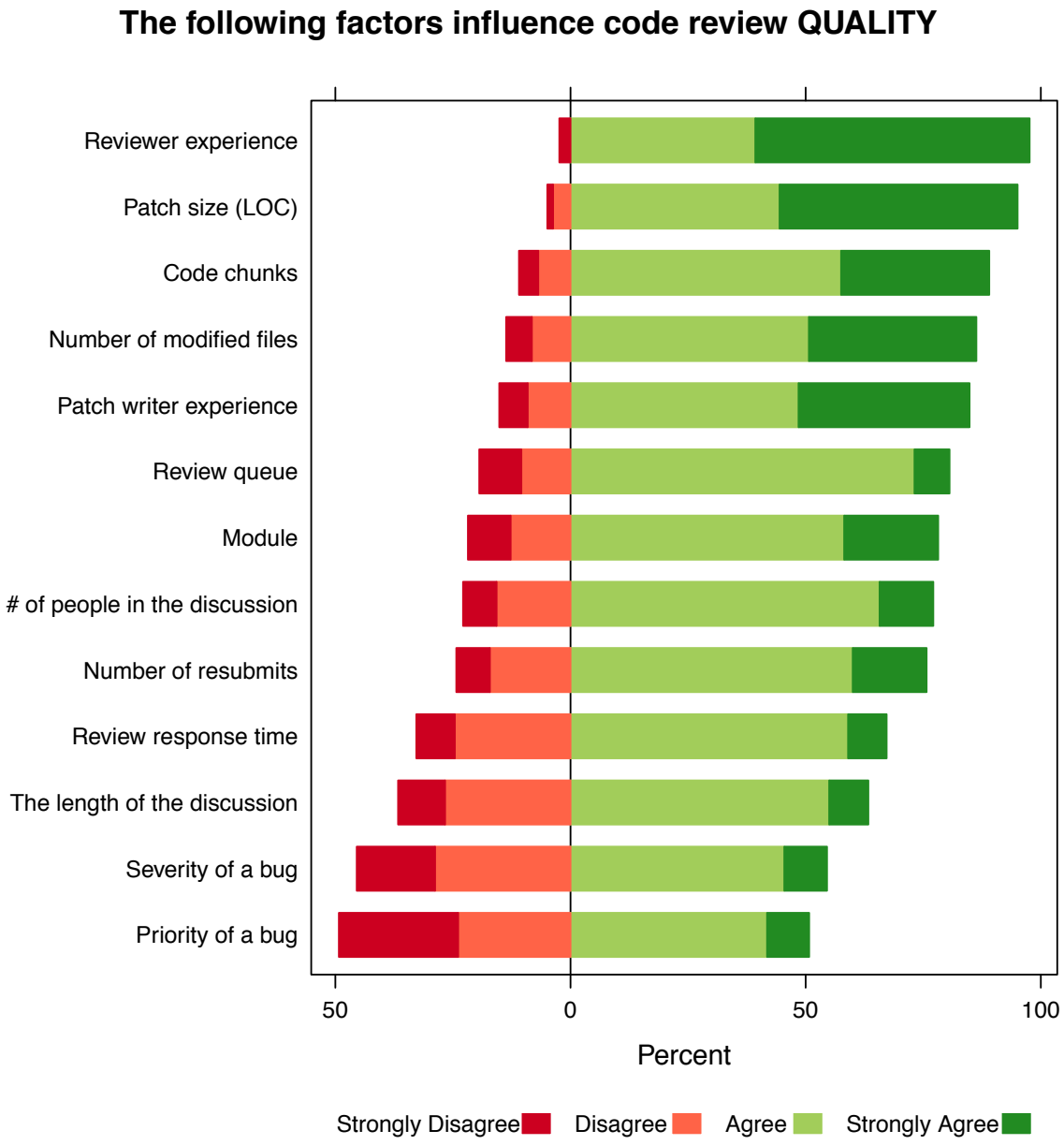

Figure 3: Q16: Factors influencing code review quality.

5 Demographics

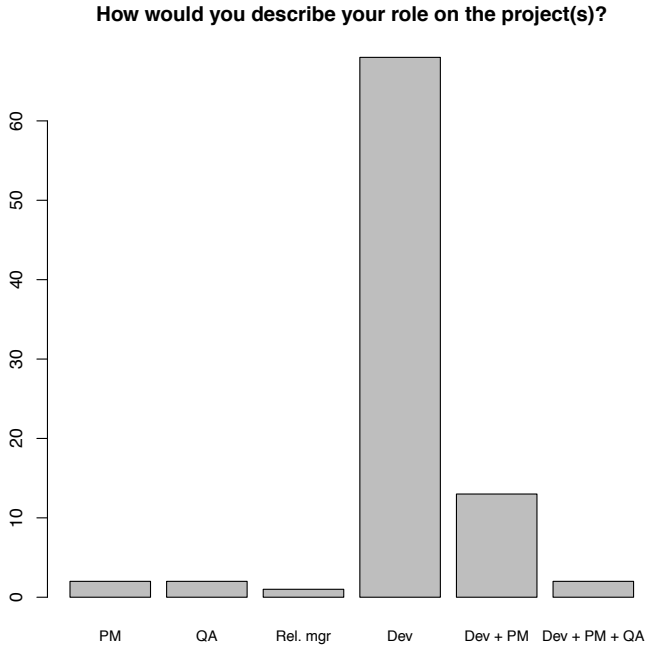

Figure 4: Q1: Role on the project.

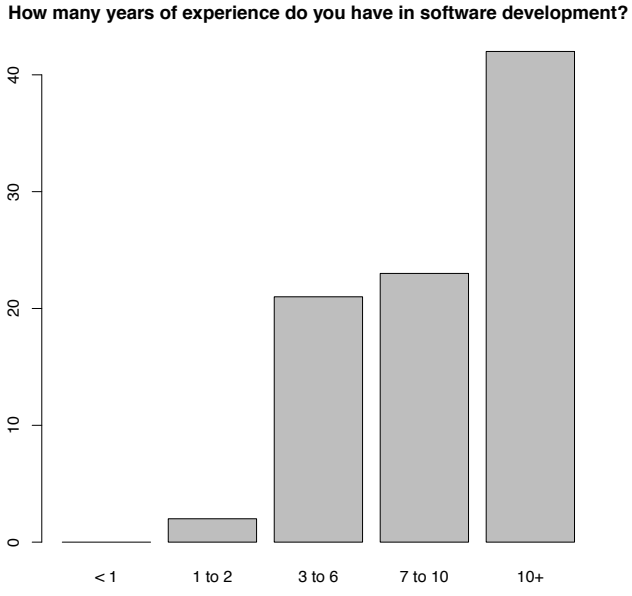

Figure 5: Q2: Years of experience in software development.

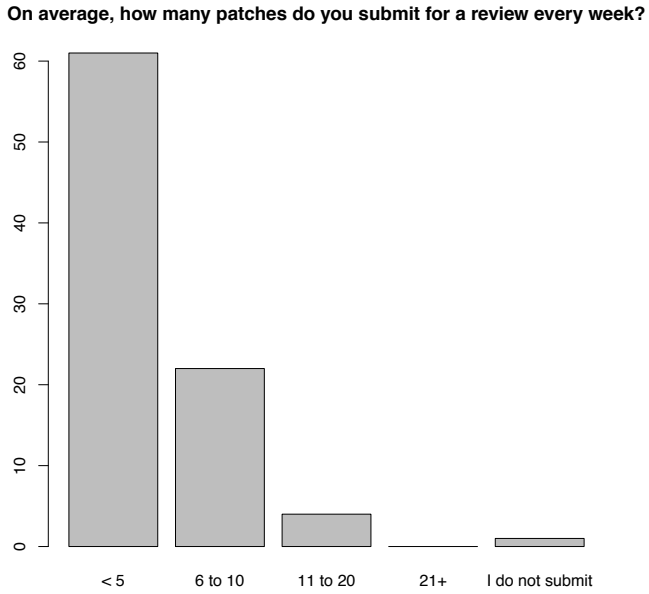

Figure 6: Q5: Submitted patches per week.

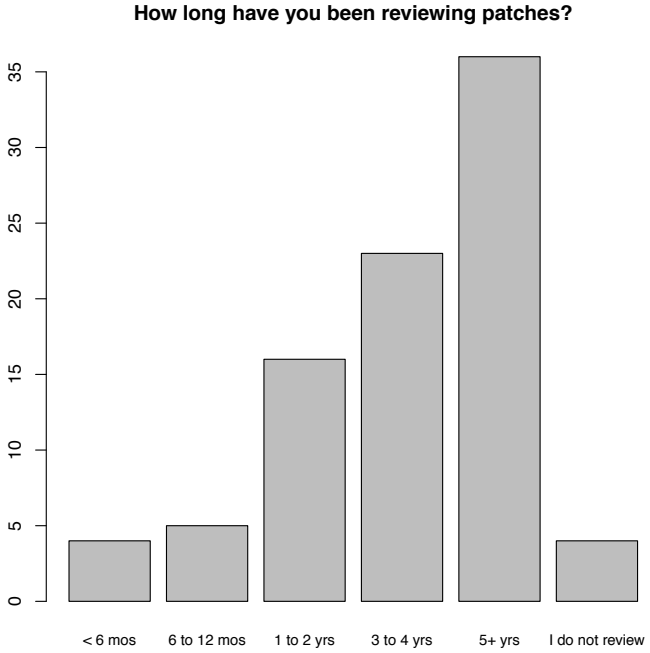

Figure 7: Q6: Reviewing experience.

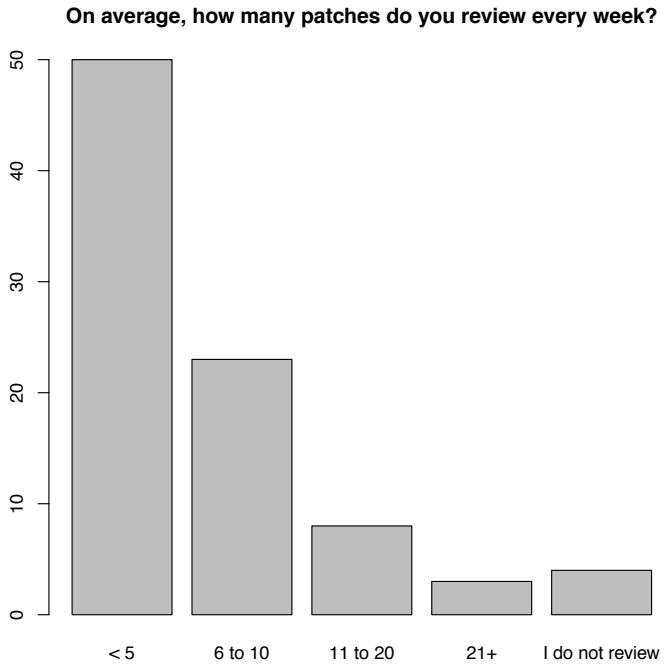

Figure 8: Q7: Number of reviews per week.

A Study Investigating Code Review Quality

*Required

1. **1) How would you describe your role on the project(s)? ***
   Check all that apply.
   Tick all that apply.

   - [ ] Software Developer/Engineer
   - [ ] Project Manager/Lead
   - [ ] QA/Testing Engineer
   - [ ] Other: .....

2. **2) How many years of experience do you have in software development? ***
   Mark only one oval.

   - ◯ < 1
   - ◯ 1 to 2
   - ◯ 3 to 6
   - ◯ 7 to 10
   - ◯ 10+

3. **3) You work for: ***
   Mark only one oval.

   - ◯ Mozilla
   - ◯ Other: .....

4. **4) How are you involved in code review? ***
   Tick all that apply.

   - [ ] Writing patches
   - [ ] Reviewing patches
   - [ ] Discussing patches/bugs
   - [ ] Other: .....

5. **5) On average, how many patches do you submit for a review every week?** *
   Mark only one oval.

   - ◯ < 5
   - ◯ 6 to 10
   - ◯ 11 to 20
   - ◯ 21+
   - ◯ I do not submit

6. **6) How long have you been reviewing patches?** *
   Mark only one oval.

   - ◯ less than 6 months
   - ◯ 6 to 12 months
   - ◯ 1 to 2 years
   - ◯ 3 to 4 years
   - ◯ 5+ years
   - ◯ I do not review

7. **7) On average, how many patches do you review every week?** *
   Mark only one oval.

   - ◯ < 5
   - ◯ 6 to 10
   - ◯ 11 to 20
   - ◯ 21+
   - ◯ I do not review

8. **8) In what environment do you typically conduct code review?**
   Mark only one oval.

   - ◯ Issue tracking (e.g., Bugzilla)
   - ◯ Copy a patch locally into editor/IDE
   - ◯ Other: .....

9. **9) Where do you discuss patches?** *
   Tick all that apply.

   - [ ] Issue tracking
   - [ ] Email
   - [ ] IRC
   - [ ] Skype/Hangouts
   - [ ] Face-to-face discussions
   - [ ] Other: .....

10. **10) The following factors influence code review DECISIONS (i.e., whether you accept/reject the patch itself after having reviewed it):** *
    Mark only one oval per row.

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Patch size (LOC)	◯	◯	◯	◯	◯
Code chunks	◯	◯	◯	◯	◯
Number of modified files	◯	◯	◯	◯	◯
Module	◯	◯	◯	◯	◯
Priority of a bug	◯	◯	◯	◯	◯
Severity of a bug	◯	◯	◯	◯	◯
Number of previous patches (resubmits)	◯	◯	◯	◯	◯
Review queue (aka load)	◯	◯	◯	◯	◯
Reviewer experience	◯	◯	◯	◯	◯
Patch writer experience	◯	◯	◯	◯	◯
Number of people involved in the discussion of a patch	◯	◯	◯	◯	◯
The length of the discussion of a patch	◯	◯	◯	◯	◯

11. **11) In your opinion, what other factors affect code review DECISIONS?**

.....

.....

.....

.....

.....

12. **12) The following factors influence code review TIME (duration):** *
Mark only one oval per row.

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Patch size (LOC)					
Code chunks					
Number of modified files					
Module					
Priority of a bug					
Severity of a bug					
Number of previous patches (resubmits)					
Review queue (aka load)					
Reviewer experience					
Patch writer experience					
Number of people involved in the discussion of a patch					
The length of the discussion of a patch					

13. **13) In your opinion, what other factors affect code review TIME?**

14. **14) How do you assess the quality of a patch?** *

15. **15) In your opinion, what characteristics do contribute to a well-done code review?** *

16. **16) The following factors influence code review QUALITY (e.g., the likelihood of detecting problems with a patch):** *

Mark only one oval per row.

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Patch size (LOC)					
Code chunks					
Number of modified files					
Module					
Priority of a bug					
Severity of a bug					
Number of previous patches (resubmits)					
Review queue (aka load)					
Reviewer experience					
Patch writer experience					
Number of people involved in the discussion of a patch					
The length of the discussion of a patch					
Review response time					

17. **17) In your opinion, what other factors affect code review QUALITY?**

18. **18) What is your biggest challenge in performing code review tasks?**

19. **19) What tools would you like to have to assist you with code review activities?**

20. **20) If you want us to keep you updated on the results of this study, please provide your email.**

References

[1] D. Freelon. ReCal2: Reliability for 2 coders. http://dfreelon.org/utils/recalfront/recal2/.

[2] Mozilla. BMO/ElasticSearch. https://wiki.mozilla.org/BMO/ElasticSearch.