Code Reviews, Software Inspections, and Code Walkthroughs: Systematic Mapping Study of Research Topics

Ilenia Fronza[1], Arto Hellas[2], Petri Ihantola[2], and Tommi Mikkonen[2]

[1]Free University of Bozen-Bolzano, Bozen-Bolzano, Italy
[2]University of Helsinki, Helsinki, Finland
ilenia.fronza@unibz.it, arto.hellas@helsinki.fi,
petri.ihantola@helsinki.fi, tommi.mikkonen@helsinki.fi

Abstract. Code reviews have been used to improve code quality since the 1970s. Most practitioners in the field of software have some experience with respect to the technique. In this mapping study we illustrate what kinds of research questions are addressed in code review literature. The following themes emerged from analysis of 75 original articles: 1) description or comparison of different code review practices, 2) behavior of reviewers (e.g., eye tracking studies), 3) communication and teamwork, 4) outcomes of code reviews (e.g., what kinds of problems are identified), 5) how properties of code to be reviewed affect reviewing, and 6) reasons for conducting code reviews. About half of the studies have been conducted with students and novices. The numbers of industry papers has significantly increased when compared to the previous reviews in the field.
Keywords. Code reviews, software inspections, code walkthroughs, mapping study.

1 Introduction

Software engineering has evolved significantly during the last decades. Code review (sometimes called as peer review) is one of the few activities surviving this evolution. Different forms of peer reviewing have been around since the 1970s [1], and practically every software engineer is familiar with the techniques, at least to some extent. Based on the previous literature, the driving force to do code reviews include finding defects, improving code quality, finding alternative solutions, transferring knowledge and improving teams awareness [2].

Code review, or manual inspection of software quality in general are widely studied topics, but systematic literature studies in the field are still rare. Brykczynski [3] conducted a literature review on checklist based quality assurance of software artifacts (i.e., requirements, design, code, testing, documentation, and process). Authors point out, that although some checks should be automated, all the listed phases of software development are likely to benefit from manual (possibly computer assisted) inspection. More recently, Ebad [4] did a systematic literature review to compare multiple manual inspection approaches and

conclude that "the most effective reading techniques in requirements, design, and coding phases are perspective-based reading, usage-based reading, and tool-assisted reading, respectively". Authors also concluded that most research seems to be based on data collected from academic, instead of industrial context.

In addition to being scarce, previous literature studies on code reviews have focused on topics that are quite specific. We argue that software engineering community would benefit from a broader understanding of themes covered in the code review literature. Therefore, the main objective of this study is to shed light on which themes can be identified in the research of code reviews. The exact research questions answered here is "what's the *focus* of the existing empirical articles on code reviews, software inspections, and code walkthroughs". In the rest of this article, we use the term *code review* to refer different review approaches (e.g., code reviews, software inspections, and code walkthroughs) We will answer our research question by conducting a systematic mapping study.

The goal of this mapping study, following the guidelines in [5], is to provide an overview of empirical research on code reviews, and identify the quantity and type of research, and results available within it. The results of this study can identify areas suitable for conducting systematic literature reviews [6], and areas where a primary study is more appropriate.

The rest of this paper is structured as follows. Section 2 gives an overview to our data collection. Section 3 presents our results. Section 4 provides an extended discussion regarding our findings and lists directions for future work. Section 5 describes limitations of this research. Finally, towards the end of the paper, Section 6 draws some conclusions.

2 Data Collection

2.1 Search for Primary Studies

The first phase of the study consisted of identifying primary studies from scientific databases. Search queries were defined to retrieve the initial selection of works to be filtered and screened later on. The search was carried out in the following digital libraries: IEEE (http://ieeexplore.ieee.org/Xplore/home.jsp), ACM (http://dl.acm.org/), and Scopus (https://www.scopus.com/).

The format of queries differs between platforms. Table 1 shows how queries were defined in each library. The actual search was conducted in June 2018; the Results-column in Table 1 indicates the number of hits in each platform.

In total, the queries produced a combined total of 1426 articles. This initial set of articles was then examined to remove duplicate articles. A total of 281 articles were removed automatically based on duplicate DOIs, leaving 1145 articles for further analysis.

2.2 Screening of Papers for Inclusion and Exclusion

The 1145 articles were screened based on their titles and abstracts. The application of inclusion and exclusion criteria to titles and abstracts was conducted by

Table 1. Executed queries for each digital library.

Database	Search	Results
IEEE	METADATA ONLY: (((“code review”) OR “code inspection”) OR “code walkthrough”)	297
ACM	acmdlTitle:(“code review” “code inspection” “code walkthrough”) OR recordAbstract:(“code review” “code inspection” “code walkthrough”) OR keywords.author.keyword:(“code review” “code inspection” “code walkthrough”)	267
Scopus	(TITLE-ABS-KEY(“code review”) OR TITLE-ABS-KEY(“code inspection”) OR TITLE-ABS-KEY(“code walkthrough”)) AND (LIMIT-TO (DOCTYPE, “cp”) OR LIMIT-TO (DOCTYPE, “ar”) OR LIMIT-TO (DOCTYPE, “re”)) AND (LIMIT-TO (SUBJAREA, “COMP”) OR LIMIT-TO (SUBJAREA, “ENGI”)) AND (LIMIT-TO (LANGUAGE, “English”))	862
Total		1426

four researchers working in parallel. We were inclusive taking a paper to full-text reading when in doubt. The following criteria state when a study was excluded:

- Studies presenting tools that are not specifically about code reviews (e.g., tools for clone detection, tools for static code analysis).
- Studies not presented in English.
- Studies presenting summaries of conferences/editorials.
- Studies not accessible in full-text.
- Books and gray literature.
- Studies that are replications of other studies.

This exclusion phase led to removal of 622 articles.

2.3 Selecting Empirical Studies in the Software Engineering Field

After the initial screening and exclusion of non-relevant articles, 523 articles remained. These articles were then further inspected, including only articles that were from the field of software engineering, and provided empirical results related to code reviews. More precisely, the following inclusion criteria was applied: 1) Studies are in the field of software engineering, and 2) Studies present empirical results on code reviews (e.g. qualitative or quantitative data on code reviews). Opinion pieces without any data or where the focus was on something else than code reviews were excluded. This led to a final data set of 75 articles.

Most of the articles in our final data set have been published in conference proceedings (61%). 32% of the publications were journal articles and the remaining 7% were published in magazines and workshops.

3 Qualitative content analysis of research questions

For all the papers, we extracted excerpts describing the objectives, hypothesis and explicit research questions of the work. Some of the papers did not have any research questions and in some cases even the objectives were loosely defined. In these cases, we extracted excerpts from the conclusions that illustrated outcomes of the work. As the last option, if we failed to find illustrative excerpts, the main contributions of a paper we defined by our own words. In addition, for every paper, we extracted the context of code reviews (e.g. academic, industry, or open source software development). Some papers had multiple contexts, in which case all were recorded.

Next, we carried out a qualitative content analysis of research questions to identify themes of the research. First, one of the authors went through all the excerpts, and created a list of potential categories. At this phase, some of the excerpts were augmented by reading the original publication. Next, themes were discussed with an another author who also read the excerpts. This resulted in some themes to be merged and some new themes to be created. Finally, when all the papers were classified to potentially multiple categories, the categorization was finalized jointly by all the authors.

The final categories and the number of articles assigned to each category are the following. The list of papers in each category divided by the context is provided in Table 2; in our study, an article could be mapped to multiple categories.

Review methods This category is related to description, combination and comparison of different code review practices. Some of the studies are descriptive and focus on explaining (new) code review methods and best practices, as in [7]: "Building on the existing literature, here we add insights from a recent large-scale study of Microsoft developers code review practices to summarize the challenges that code-change authors and reviewers face, suggest best code-reviewing practices, and discuss tradeoffs that practitioners should consider." Some papers in the category focus on (quantitative) comparison review methods as illustrate in the following excerpt of research hypothesis [8]: "There is significant difference in the number defects found by those subjects performing ad-hoc inspection and those performing systematic inspection of object oriented code". (33 papers)

Human behaviour Papers in this category focus on Individual differences between reviewers (e.g., "how effective developers are at conducting code reviews and the degree of variation among them" [9] and behaviour of individual reviewer, e.g., by using eye-tracking [10, 11]. The category is often linked to the previous category of describing new review method and research questions such as "Do developers who are shown information that could potentially help avoid the introduction of bugs behave differently than without that information" [12] (15 papers)

Teamwork Papers in this category focus on communication, team configurations and teamwork. Examples of research questions in this category are

"Does the number of involved teams influence the effectiveness of distributed code review" [13] and "what do reviewers discuss in test code reviews" [14] In many cases, the role of teamwork was implicit while research questions were more broadly defined. (14 papers)

Outcome of code reviews Papers in this category focus on effect of code reviews, for example, which kinds of errors are found or how many of the errors can be found. The category is closely linked to *review methods* category. Examples of research questions in category include "What is the impact of continuous code reviews and inspections on code quality, What are the most common bugs among the code written by sophomores? What are the most common code smells identified within the code written by sophomores?" [15] (14 papers)

Role of code to be reviewed Papers in this category focus on the relation of reviews and code to be reviewed. How properties of code to be reviewed affect reviewing is more specific when compared to other categories, but it was still clearly emerging from the data. Excepts illustrating this category include "Does the number lines of code to be reviewed influence the effectiveness of distributed code review?" [13] and "What factors can influence how long it takes for a patch to be reviewed?" [16] (5 papers)

Reasons for conducting code reviews Papers in this category address explicitly the reasons or motivations for conducting code reviews, with research questions like "What are the motivations for code review at Google [...] How do Google developers perceive code review?" [17] (4 papers)

Table 2. Topical classification of papers. (*) Some papers use data from more than one context, for instance when comparing industry with academia.

Category	n	Context*	Papers
Review methods	33	industry (14), open source (3), academia (15), unclear (1), government (1), simulated review (1)	[1, 7, 8, 14, 17–45]
Human behaviour	15	industry (5), open source (3), academia (8)	[9–12, 46–56]
Teamwork	14	industry (9), open source (2), academia (3)	[7, 13, 14, 16, 24, 35, 57–64]
Outcome of code reviews	14	industry (5), open source (7), academia (2)	[15, 43, 49, 65–75]
Role of code to be reviewed	5	industry (1), open source (4)	[13, 14, 16, 76, 77]
Reasons for conducting code reviews	4	industry (2), academia (1), not defined (1)	[17, 78–80]

While reviews are considered somewhat classical technique, we identified only few studies published prior to mid-1990s. Publication year of the studies are illustrated in Figure 1.

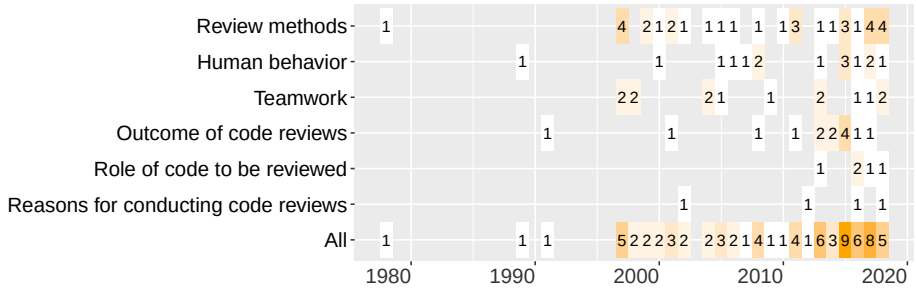

Fig. 1. The number of publication per year for each category (a single paper can be classified into multiple categories) and the total number of publications identified each year.

4 Discussion

The largest category of research topics identified in this mapping study, with 38% of papers related to it is description, combination and comparison of different code review practices. The category overlaps heavily with other categories, however. For example, description of a new code review approach is often related to analysis of how people behave when applying the new methods, and quantifying the results of code reviews.

Indeed, the second largest groups in our analysis are "Communication, team configurations and teamwork", " Effect of code reviews (e.g., which kinds of errors are found)", and "Individual differences and behavior of reviewers". Each of these themes to related to ca. 16% of all the papers.

Although we were able to identify many studies that either compare (effectiveness of) code review techniques or identify benefits of code review, there are only few studies that compare effectiveness of code reviews to other quality assurance techniques (e.g., [75]). We would like to see more studies that compare code reviews directly to alternative or complementary methods, such as test driven development or pair programming.

Three different contexts are common when considering reviews that contain empirical results – open source, industry, and education. Sometimes, they (partially) overlap or complement each other in the studies (hence the small mismatch in numbers per category and total in Table 2). In our study, ca. half of the studies have been conducted with students and novices. This is a significant improvement to earlier review by Ebad [4], where only four percent of the research was conducted in the industry context.

We are somewhat surprised by the fact that there are very few old articles with empirical evidence, although code reviews is considered a classic topic. Furthermore, these initial papers were really placing the focus on the essentials of code reviews as a quality assurance and bug-fixing instrument. Based on our study, mid-1990s seem to mark the point when there was increasing empirical interest in reviews in general, and only after 2010 there are several papers that

empirically study reviews. Granted, the increasing interest in reviews has also meant that the research is more versatile, addressing various topics revolving around reviews but not necessarily studying their effectiveness as a mechanism for quality assurance.

Finally, we did not consider the evolution of the term code review in this study; however it is clear that the meaning of the term has evolved significantly. For example, in 1980s, the term software review meant software inspection where the quality of a software module was inspected following a certain process [81], whereas today software review more often refers to the acceptance decision for inclusion of a contribution in an open source project [82]. Understanding this evolution is a topic for future work.

5 Limitations and Threats to Validity

There are multiple biases related to the selection of primary research. Retrospectively, the biggest selection choice we made in the paper was to exclude tool papers, where empirical evidence was focused on evaluating the tool. It is likely that in the process we also eliminated some data that also acts as evidence regarding code reviews more generally as well. By inspecting many of the tools papers as well, we came to a conclusion that in general they are not comparable to papers that are solely dedicated to code reviews. However, studying the tool perspective remains an possible direction for future work.

Another dimension we deliberately excluded in this study is the use of code reviews in education. Such studies take a very different stand to code reviews, and while they would have contributed to versatility of the mapping study in general, based on reading many of them, they might have resulted in a category of their own in the classification. Hence, we feel that they deserve a paper of their own. Publications that used educational context, but did not focus on how to teach code reviews were included, however.

The general limitations associated with any mapping study also apply to our work, including in particular bias in selection of the reviewed papers and inaccuracy in data extraction. Since we mainly relied on search engines to retrieve the primary studies, the search engines may have influenced the completeness of the identified studies. The extraction process may have also resulted in inaccuracies, even though the reviewers practiced extraction jointly. Quality assessment of studies in systematic reviews still remains a major problem [83].

6 Conclusions

Code inspections are a classic approach to quality assurance. Despite frequent use of the method in industry, there are only few systematic literature studies of the field. In this systematic mapping study we have illustrated what kinds of research themes can be identified in the code review literature.

The following themes emerged from analysis of 75 original articles: 1) description or comparison of different code review practices, 2) human behavior

and differences between individual reviewers 3) communication and teamwork, 4) outcomes of code reviews, 5) how properties of code to be reviewed affect reviewing, and 6) reasons for conducting code reviews.

While many of the papers identified in this survey address effectiveness of code reviews, comparisons between code review to other approaches aiming to improve software quality are uncommon. Moreover, much of the knowledge is at least partially outdated due to the changes in software development and deployment methodologies. In contrast, softer issues, such as team behavior and participants roles in code review, have been gaining traction in research, resulting in various studies of code reviews from the socio-technical dimension.

While we acknowledge the importance of the socio-technical dimension, we believe that there is a need for further primary studies from purely technical point of view, taking code reviews as a quality assurance technique back into focus.

References

1. Myers, G.J.: A controlled experiment in program testing and code walkthroughs/inspections. Communications of the ACM **21**(9) (1978) 760–768
2. Bacchelli, A., Bird, C.: Expectations, outcomes, and challenges of modern code review. In: Proceedings of the 2013 international conference on software engineering, IEEE Press (2013) 712–721
3. Brykczynski, B.: A survey of software inspection checklists. ACM SIGSOFT Software Engineering Notes **24**(1) (1999) 82
4. Ebad, S.: Inspection reading techniques applied to software artifacts-a systematic review. Computer Systems Science and Engineering **32**(3) (2017) 213–226
5. Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M.: Systematic mapping studies in software engineering. In: Proc. of the 12th Int. Conf. on Evaluation and Assessment in Software Engineering. EASE'08, BCS Learning & Development (2008) 68–77
6. Kitchenham, B.A., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Tech. Report EBSE-2007-01, Keele Univ. (2007)
7. Greiler, M., Bird, C., Storey, M.A., MacLeod, L., Czerwonka, J.: Code reviewing in the trenches: Understanding challenges, best practices and tool needs. (2016)
8. Dunsmore, A., Roper, M., Wood, M.: Systematic object-oriented inspectionan empirical study. In: Proc. of the 23rd Int. Conf. on Softw. Eng., IEEE (2001) 135–144
9. Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A., Wagner, D.: An empirical study on the effectiveness of security code review. In: International Symposium on Engineering Secure Software and Systems, Springer (2013) 197–212
10. Begel, A., Vrzakova, H.: Eye movements in code review. In: Proc. of the Workshop on Eye Movements in Programming, ACM (2018) 5
11. Uwano, H., Nakamura, M., Monden, A., Matsumoto, K.i.: Analyzing individual performance of source code review using reviewers' eye movement. In: Proc. of the 2006 symposium on Eye tracking research & applications, ACM (2006) 133–140
12. Foss, S.L., Murphy, G.C.: Do developers respond to code stability warnings? In: Proc. of the 25th Annual Int. Conf. on Computer Science and Software Engineering, IBM Corp. (2015) 162–170

13. dos Santos, E.W., Nunes, I.: Investigating the effectiveness of peer code review in distributed software development. In: Proc. of the 31st Brazilian Symposium on Software Engineering, ACM (2017) 84–93
14. Spadini, D., Aniche, M., Storey, M.A., Bruntink, M., Bacchelli, A.: When testing meets code review: why and how developers review tests. In: 2018 IEEE/ACM 40th Int. Conf. on Softw. Eng. (ICSE), IEEE (2018) 677–687
15. Sripada, S.K., Reddy, Y.R.: Code comprehension activities in undergraduate software engineering course-a case study. In: 2015 24th Australasian Software Engineering Conference, IEEE (2015) 68–77
16. Baysal, O., Kononenko, O., Holmes, R., Godfrey, M.W.: The influence of non-technical factors on code review. In: 2013 20th Working Conference on Reverse Engineering (WCRE), IEEE (2013) 122–131
17. Sadowski, C., Söderberg, E., Church, L., Sipko, M., Bacchelli, A.: Modern code review: a case study at google. In: Proc. of the 40th Int. Conf. on Softw. Eng.: Software Engineering in Practice, ACM (2018) 181–190
18. Fracz, W., Dajda, J.: Experimental validation of source code reviews on mobile devices. In: Int. Conf. on Computational Science and Its Applications, Springer (2017) 533–547
19. Baum, T., Leßmann, H., Schneider, K.: The choice of code review process: A survey on the state of the practice. In: Int. Conf. on Product-Focused Software Process Improvement, Springer (2017) 111–127
20. Ferreira, A.L., Machado, R.J., Silva, J.G., Batista, R.F., Costa, L., Paulk, M.C.: An approach to improving software inspections performance. In: 2010 IEEE Int. Conf. on Software Maintenance, IEEE (2010) 1–8
21. Vassallo, C., Panichella, S., Palomba, F., Proksch, S., Zaidman, A., Gall, H.C.: Context is king: The developer perspective on the usage of static analysis tools. In: IEEE 25th Int. Conf. on Softw. Analysis, Evol. and Reengineering (SANER), IEEE (2018) 38–49
22. Höst, M., Johansson, C.: Evaluation of code review methods through interviews and experimentation. J. Syst. Softw. **52**(2-3) (2000) 113–120
23. Kamsties, E., Lott, C.M.: An empirical evaluation of three defect-detection techniques. In Schäfer, W., Botella, P., eds.: Software Engineering — ESEC '95, Berlin, Heidelberg, Springer Berlin Heidelberg (1995) 362–383
24. Müller, M.M.: Are reviews an alternative to pair programming? Empir. Softw. Eng. **9**(4) (2004) 335–351
25. Khandelwal, S., Sripada, S.K., Reddy, Y.R.: Impact of gamification on code review process: An experimental study. In: Proc. of the 10th Innovations in Software Engineering Conference, ACM (2017) 122–126
26. Hatton, L.: Testing the value of checklists in code inspections. IEEE software **25**(4) (2008) 82–88
27. Belli, F., Crisan, R.: Empirical performance analysis of computer-supported code-reviews. In: Proceedings The Eighth International Symposium on Software Reliability Engineering, IEEE (1997) 245–255
28. El Emam, K., Laitenberger, O.: Evaluating capture-recapture models with two inspectors. IEEE Trans. Softw. Eng. **27**(9) (2001) 851–864
29. Hirao, T., Ihara, A., Matsumoto, K.i.: Pilot study of collective decision-making in the code review process. In: Proc. of the 25th Annual Int. Conf. on Computer Science and Software Engineering, IBM Corp. (2015) 248–251
30. Olorisade, B.K., Vegas, S., Juristo, N.: Determining the effectiveness of three software evaluation techniques through informal aggregation. Information and software technology **55**(9) (2013) 1590–1601

31. Runeson, P., Stefik, A., Andrews, A., Gronblom, S., Porres, I., Siebert, S.: A comparative analysis of three replicated experiments comparing inspection and unit testing. In: 2011 Second International Workshop on Replication in Empirical Software Engineering Research, IEEE (2011) 35–42
32. De Vreede, G.J., Koneri, P.G., Dean, D.L., Fruhling, A.L., Wolcott, P.: A collaborative software code inspection: the design and evaluation of a repeatable collaboration process in the field. International Journal of Cooperative Information Systems **15**(02) (2006) 205–228
33. Hémeury, B.: Report on the vera experiment. In: Int. Conf. on Reliable Software Technologies, Springer (1999) 103–113
34. Kelly, D., Shepard, T.: Qualitative observations from software code inspection experiments. In: Proc. of the 2002 conference of the Centre for Advanced Studies on Collaborative research, IBM Press (2002) 5
35. Porter, A.A., Siy, H.P., Toman, C.A., Votta, L.G.: An experiment to assess the cost-benefits of code inspections in large scale software development. IEEE Trans. Softw. Eng. **23**(6) (1997) 329–346
36. Wang, Y.Q., Qi, Z.Y., Zhang, L.J., Song, M.J.: Research and practice on education of SQA at source code level. Int. J. of Eng. Education **27**(1) (2011) 70
37. Panko, R.R.: Applying code inspection to spreadsheet testing. Journal of Management Information Systems **16**(2) (1999) 159–176
38. Koneri, P.G., de Vreede, G.J., Dean, D.L., Fruhling, A.L., Wolcott, P.: The design and field evaluation of a repeatable collaborative software code inspection process. In: Int. Conf. on Collaboration and Technology, Springer (2005) 325–340
39. Cristia, M., Frydman, C.: Formal and semi-formal verification of a web voting system. International Journal of Web Information Systems **11**(2) (2015) 183–204
40. da Silva Neto, A.V., Vismari, L.F., Gimenes, R.A.V., Sesso, D.B., de Almeida, J.R., Cugnasca, P.S., Camargo, J.B.: A practical analytical approach to increase confidence in pld-based systems safety analysis. IEEE Systems J. (99) (2017) 1–12
41. Wood, M., Roper, M., Brooks, A., Miller, J.: Comparing and combining software defect detection techniques: a replicated empirical study. In: Software EngineeringESEC/FSE'97. Springer (1997) 262–277
42. Wilkerson, J.W., Nunamaker, J.F., Mercer, R.: Comparing the defect reduction benefits of code inspection and test-driven development. IEEE Trans. Softw. Eng. **38**(3) (2011) 547–560
43. Morales, R., McIntosh, S., Khomh, F.: Do code review practices impact design quality? a case study of the qt, vtk, and itk projects. In: IEEE 22nd Int. Conf. on Softw. Analysis, Evol., and Reengineering (SANER), IEEE (2015) 171–180
44. Oliveira, R., Estácio, B., Garcia, A., Marczak, S., Prikladnicki, R., Kalinowski, M., Lucena, C.: Identifying code smells with collaborative practices: A controlled experiment. In: 2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), IEEE (2016) 61–70
45. Swamidurai, R., Dennis, B., Kannan, U.: Investigating the impact of peer code review and pair programming on test-driven development. In: IEEE SOUTHEASTCON 2014, IEEE (2014) 1–5
46. Bisant, D.B., Lyle, J.R.: A two-person inspection method to improve programming productivity. IEEE Trans. Softw. Eng. (10) (1989) 1294–1304
47. McMeekin, D.A., von Konsky, B.R., Chang, E., Cooper, D.J.: Measuring cognition levels in collaborative processes for software engineering code inspections. In: Int. Conf. on IT Revolutions, Springer (2008) 32–43

48. McMeekin, D.A., von Konsky, B.R., Chang, E., Cooper, D.J.: Checklist based reading's influence on a developer's understanding. In: 19th Australian Conference on Software Engineering (aswec 2008), IEEE (2008) 489–496
49. McIntosh, S., Kamei, Y., Adams, B., Hassan, A.E.: An empirical study of the impact of modern code review practices on software quality. Empir. Softw. Eng. **21**(5) (2016) 2146–2189
50. Stålhane, T., Awan, T.H.: Improving the software inspection process. In: European Conference on Software Process Improvement, Springer (2005) 163–174
51. Dunsmore, A., Roper, M., Wood, M.: The role of comprehension in software inspection. J. Syst. Softw. **52**(2-3) (2000) 121–129
52. Da Cunha, A.D., Greathead, D.: Does personality matter?: an analysis of code-review ability. Communications of the ACM **50**(5) (2007) 109–112
53. Thongtanunam, P., McIntosh, S., Hassan, A.E., Iida, H.: Investigating code review practices in defective files: An empirical study of the qt system. In: Proc. of the 12th Working Conference on Mining Software Repositories, IEEE (2015) 168–179
54. Kononenko, O., Baysal, O., Guerrouj, L., Cao, Y., Godfrey, M.W.: Investigating code review quality: Do people and participation matter? In: 2015 IEEE Int. Conf. on software maintenance and evolution (ICSME), IEEE (2015) 111–120
55. de Mello, R.M., Oliveira, R.F., Garcia, A.F.: On the influence of human factors for identifying code smells: a multi-trial empirical study. In: 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE (2017) 68–77
56. Murakami, Y., Tsunoda, M., Uwano, H.: Wap: Does reviewer age affect code review performance? In: 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), IEEE (2017) 164–169
57. Seaman, C.B., Basili, V.R.: An empirical study of communication in code inspections. In: Proc. of the (19th) Int. Conf. on Softw. Eng., IEEE (1997) 96–106
58. Porter, A., Siy, H., Mockus, A., Votta, L.: Understanding the sources of variation in software inspections. ACM Trans. Softw. Eng. Methodol. **7**(1) (1998) 41–79
59. Müller, M.M.: Two controlled experiments concerning the comparison of pair programming to peer review. J. Syst. Softw. **78**(2) (2005) 166–179
60. Spohrer, K., Kude, T., Schmidt, C.T., Heinzl, A.: Knowledge creation in information systems development teams: The role of pair programming and peer code review. In: ECIS. (2013) 213
61. Miller, J., Yin, Z.: A cognitive-based mechanism for constructing software inspection teams. IEEE Trans. Softw. Eng. **30**(11) (2004) 811–825
62. Sutherland, A., Venolia, G.: Can peer code reviews be exploited for later information needs? In: 2009 31st Int. Conf. on Softw. Eng.-Companion Volume, IEEE (2009) 259–262
63. Seaman, C.B., Basili, V.R.: Communication and organization: An empirical study of discussion in inspection meetings. IEEE Trans. Softw. Eng. **24**(7) (1998) 559–572
64. Bosu, A., Carver, J.C., Bird, C., Orbeck, J., Chockley, C.: Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft. IEEE Trans. Softw. Eng. **43**(1) (2016) 56–75
65. Panichella, S., Arnaoudova, V., Di Penta, M., Antoniol, G.: Would static analysis tools help developers with code reviews? In: IEEE 22nd Int. Conf. on Softw. Analysis, Evol., and Reengineering (SANER), IEEE (2015) 161–170

66. Thompson, C., Wagner, D.: A large-scale study of modern code review and security in open source projects. In: Proc. of the 13th Int. Conf. on Predictive Models and Data Analytics in Software Engineering, ACM (2017) 83–92
67. Lei, Q., He, Z., Fuqun, H., Bin, L.: Classification of air on-board software code defects and investigations. Procedia Engineering **15** (2011) 3577–3583
68. Russell, G.W.: Experience with inspection in ultralarge-scale development. IEEE software **8**(1) (1991) 25–31
69. Siy, H., Votta, L.: Does the modern code inspection have value? In: Proc. of the IEEE Int. Conf. on Softw- Maintenance (ICSM'01), IEEE (2001) 281
70. Bavota, G., Russo, B.: Four eyes are better than two: On the impact of code reviews on software quality. In: 2015 IEEE Int. Conf. on Software Maintenance and Evolution (ICSME), IEEE (2015) 81–90
71. Bosu, A., Carver, J.C.: Impact of peer code review on peer impression formation: A survey. In: 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, IEEE (2013) 133–142
72. Beller, M., Bacchelli, A., Zaidman, A., Juergens, E.: Modern code reviews in open-source projects: Which problems do they fix? In: Proc. of the 11th working conference on mining software repositories, ACM (2014) 202–211
73. Bernhart, M., Grechenig, T.: On the understanding of programs with continuous code reviews. In: 2013 21st Int. Conf. on Program Comprehension (ICPC), IEEE (2013) 192–198
74. Mäntylä, M.V., Lassenius, C.: What types of defects are really discovered in code reviews? IEEE Trans. Softw. Eng. **35**(3) (2008) 430–448
75. Runeson, P., Stefik, A., Andrews, A.: Variation factors in the design and analysis of replicated controlled experiments. Empir. Softw. Eng. **19**(6) (2014) 1781–1808
76. Baysal, O., Kononenko, O., Holmes, R., Godfrey, M.W.: Investigating technical and non-technical factors influencing modern code review. Empir. Softw. Eng. **21**(3) (2016) 932–959
77. Nanthaamornphong, A., Chaisutanon, A.: Empirical evaluation of code smells in open source projects: preliminary results. In: Proc. of the 1st International Workshop on Software Refactoring, ACM (2016) 5–8
78. Perry, D.E., Porter, A., Wade, M.W., Votta, L.G., Perpich, J.: Reducing inspection interval in large-scale software development. IEEE Trans. Softw. Eng. **28**(7) (2002) 695–705
79. Jenkins, G.L., Ademoye, O.: Can individual code reviews improve solo programming on an introductory course? Innovation in Teaching and Learning in Information and Computer Sciences **11**(1) (2012) 71–79
80. Baum, T., Liskin, O., Niklas, K., Schneider, K.: Factors influencing code review processes in industry. In: Proc. of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM (2016) 85–96
81. Ackerman, A.F., Buchwald, L.S., Lewski, F.H.: Software inspections: an effective verification process. IEEE software **6**(3) (1989) 31–36
82. Rigby, P., Cleary, B., Painchaud, F., Storey, M.A., German, D.: Contemporary peer review in action: Lessons from open source development. IEEE software **29**(6) (2012) 56–61
83. Kitchenham, B., Brereton, P.: A systematic review of systematic review process research in software engineering. Inf. Softw. Technol. **55**(12) (2013) 2049–2075