Modern Code Reviews: Preliminary Results of an Analysis of the State of the Art with Respect to the Role Played by Human Factors

Aygul Malikova[a] and Giancarlo Succi[b]
Innopolis University, Innopolis, Russia

Keywords: Modern Code Review, Social Interactions, Problems, Quality.

Abstract: Modern Code Reviewing has shown to be an effective mechanism to identify bugs in the code; however, given their intrinsic subjectivity, they can be significantly affected by human factors such as interpersonal relationships. This paper focuses on exploring such issues, with specific attention to social iterations and personal factors. Future work includes experimental evaluations to verify the research hypothesis related to improving the quality of the process under the study.

1 INTRODUCTION

Code reviews have been a common software engineering practice for the last four decades (Fagan, 1999). An improved version of it, Modern Code Reviews (MCR), has been proven particularly effective (Bacchelli, 2013), and it is the subject of this work.

The effects of MCR have been variously explored in the research, with experiments also done in large software companies, such as Microsoft (Bacchelli, 2013; Bosu et al., 2017; Rigby, 2013), Google (Sadowski et al., 2018; Rigby, 2013), Mozilla (Kononenko et al., 2016), and in Open Source projects (Rigby, 2013).

In such works, a significant number of problems are highlighting. For example, misunderstandings, distance, social interactions, and customization among developers (Bacchelli, 2013; Sadowski et al., 2018). And these kinds of people issues is the area of the present research. Specifically, considering how social interactions and human factors can lead to not objective and misleading reviews.

Previous research focused mainly on the problems of the modern code review and their consequences, while the possible solutions to these issues are not well-studied. In this regard, the our study aims not only at investigating social problems in MCR described above, but it also aims at providing possible ways to prevent them and improve the overall quality of Modern Code Review. To this end, our key research questions are:

1. How do social interactions affect the code reviewing process?
2. How to prevent the artefacts induced by social interactions and to improve the quality of Modern Code Review?

For this study, we are going to apply various research techniques:

- **Systematic literature review** for studying the previous related works and systematize the findings (Siddaway et al., 2019), including
- **Forward and backward snowballing** for the search process of papers (Wohlin, 2014);
- **Qualitative research approach** for collecting the data from the interviews (Bolderston, 2012);

As a methodology, we chose face-to-face semi-structural interviews with developers of different companies to investigate the topic and gather the statistics. The goal behind conducting the survey was to:

- understand how social interactions affect the code reviewing process ,
- define a strategy to prevent the artifacts induced by social and negative interactions and to improve the quality of Modern Code Review .

This work is structured as follows. Section 2 presents the background and related works review. Section 3 describes the Systematic Literature Review and section 4 an arguable opinion about the non-technical Modern Code Reviews issues. Description

[a] https://orcid.org/0000-0002-8757-5282
[b] https://orcid.org/0000-0001-8847-0186

Malikova, A. and Succi, G.
Modern Code Reviews: Preliminary Results of an Analysis of the State of the Art with Respect to the Role Played by Human Factors.
DOI: 10.5220/0010576302530260
In *Proceedings of the 16th International Conference on Software Technologies (ICSOFT 2021)*, pages 253-260
ISBN: 978-989-758-523-4

of future work presented in Section 5. And finally, the conclusion is placed in 6.

2 BACKGROUND

2.1 Modern Code Review

Modern Code review (MCR) is a multistage process where developers evaluate source code written by others to enhance the software quality (Fatima et al., 2019). The term MCR appeared recently in 2013 and represents the lightweight variant of Fagan's code (Bacchelli, 2013). Other distinguishing features of MCR from other types of review processes are tool-based approach and asynchrony (Bacchelli, 2013). Applying tool-based review assumes adapting some instrument for bringing structure to the process of reviewing patches and supporting the logistics of review (Sadowski et al., 2018). There many different tools which are used by OSS and industrial projects, for example, CodeFlow (used by Microsoft), Gerrit (Google's Chromium and OSS projects), ReviewBoard (VMware), Phabricator (Facebook), and others (Sadowski et al., 2018). Asynchrony allows participants to conduct code review independently of time and space (Stein et al., 1997).

The software development process involves individual or collaborative work on it. Its vital part, MCR, requires at least two people: the author and the reviewer. In some companies, it requires more than one reviewer, for example, VMware involves two independent reviewers (Rigby and Germán, 2005) and Microsoft required an average four people (Rigby, 2013). The flow of process consists of several steps, which are general for many different companies, such as creating, previewing, commenting, addressing feedback, and approving (Sadowski et al., 2018).

2.2 Rationale for Performing MCR

One of the most compelling reasons for performing MCR is to prevent developers from inappropriately "protect" the code that they develop (Bacchelli, 2013). "Protecting" means organizing the process to prevent none apart from them can modify or even use, in the most extreme cases, such as their code. In addition, review also amounts giving insight about the code to other developers, sharing the information across the team, supporting them, and improving the overall process and quality of code (Bacchelli, 2013).

We have then decided to perform an empirical study on the matter (Vernazza et al., 2000; Succi et al., 2001a; Succi et al., 2001b; Musílek et al., 2002; Sillitti et al., 2004; Scotto et al., 2004; Pedrycz and Succi, 2005; Ronchetti et al., 2006; Scotto et al., 2006; Moser et al., 2008b; Moser et al., 2008a; Rossi et al., 2010; Pedrycz et al., 2011; Pedrycz et al., 2012; Sillitti et al., 2012; Corral et al., 2015). A detailed empirical study conducted at Google evidenced that from the perspective of developers, the main reasons for doing MCR are education (teaching or learning), maintaining the organization norms, establishing the boundaries around source code, and finally, prevention of bugs, defects, and other quality issues (Sadowski et al., 2018).

2.3 Issues in Modern Code Reviews

There are several issues that developers face during modern code reviews. Understanding the code and the reasons for changing it is considered as the main problem and one of the hardest to solve (Bacchelli, 2013). From an interview with twelve Google developers, breakdowns concerning aspects of the process were identified, which are social interactions, distance, review subject, context, and customization. The social interactions will be explained in more details in the next paragraphs. The distance can be treated as a physically or between different teams or different roles. Review subject comes from a lack of understanding of the code. Context problem means misunderstanding of reasons for changing the code. Finally, the customization is a problem of various requirements of different companies (Sadowski et al., 2018).

Since a human is a social being and while getting in touch with other ones the communication occurs, thereby the positive and negative effects can appear in the results of working together, in this case, during the peer review. Social influences include the trust relationship between the author (Zhang et al., 2020), interaction among the MCR workforce (history of interactions, its volume, the sequence, mode, and so on) (Bosu et al., 2017; Fatima et al., 2019), relationships between the group members (Bosu et al., 2017) and the impression of the individual author or reviewer (Bosu et al., 2017; Fatima et al., 2019). Other non-technical issues influencing the code review process are the personnel factors, which are the team factors, team interaction, and reviewer response (Fatima et al., 2019). Finally, the individual factors including skill, characteristics, emotions, knowledge and experience, historical factors, psychological safety, work style, and individual biasness (Fatima et al., 2019). This work is based on examining the social aspects.

The problems that arise from social communica-

tions between developers and affect the code reviewing process are common in different kind of team: distributed and co-located (Bosu et al., 2017). But it is worth considering that with increasing the team size, the social networks become less close (Crowston and Howison, 2003). Moreover, the researchers found that a few individuals have a large number of interactions, while most have only a few (Crowston and Howison, 2003). The surveys to identify the effects of social factors were conducted with OSS and Microsoft teams. The results showed that constructs such as trust, perception of expertise, reliability and friendship have a large impact on code review processes (Bosu et al., 2017).

3 SYSTEMATIC LITERATURE REVIEW

As mentioned, we have been performed a Systematic Literature Review (SLR) for gathering a comprehensive understanding of the state of the art (Kitchenham, 2004). SLR aims to address the problems of conflicting findings by identifying, critically evaluating, and integrating the sources of all relevant, high-quality individual studies (Siddaway et al., 2019). In our SLR we have followed the steps coming from the original work of Kitchenham, 2004 (Table 1). In the remaining of this section we will detail such steps.

Table 1: Systematic literature review process (Brereton et al., 2007).

Phase 1. Plan Review	1. Specify Research Questions
	2. Develop Review Protocol
	3. Validate Review Protocol
Phase 2. Conduct Review	4. Identify Relevant Research
	5. Select Primary Studies
	6. Assess Study Quality
	7. Extract Required Data
	8. Synthesise Data
Phase 3. Document Review	9. Write Review Report
	10. Validate Report

3.1 Developing the Review Protocol

The development of the review protocol is a significant part of performing a systematic literature review, and it aims to minimize possible inconsistencies in the analysis of existing work, detailing in advance how the systematic review is to be conducted (Brereton et al., 2007). Table 2 shows the steps of this phase.

Table 2: Process to develop review protocol (Galster et al., 2014).

Step 1	Define search strategy
Step 2	Define inclusion + exclusion criteria
Step 3	Define research process
Step 4	Define quality criteria
Step 5	Design data extraction form
Step 6	Define data analysis + presentation

3.2 Search Strategy

The search for the necessary literature took place in two ways: automatic by the research string and snowballing. For setting the research string, the keywords and their alternative have to be defined.

The keywords are *modern code review, social interactions, problems, quality*.

Table 3: Keywords.

Modern Code Review	Social Interactions	Problems	Quality
Modern Code Inspection, Contemporary Code Review	Group Interactions, Human Factors	Challenges, Issues	Capacity

The search string is the following:

((”Modern code review” OR ”Modern Code Inspection” OR ”Contemporary Code Review”) AND ((”Social Interactions” OR ”Group Interactions” OR ”Human Factors”) AND (”Problems” OR ”Challenges” OR ”Issues”)) AND (”Quality” OR ”Capacity”))

Table 4 presents the results of the automatic search by the research string.

Table 4: Results of automatic search.

Database	**Number of found works**
IEEE XPlore Digital Library	10
ACM Digital Library	14
Google Scholar	214

The snowballing technique has been used to extend the search for the reviewing literature. Snow-

balling implies using the reference list of a paper or the citations to the studies to identify additional sources (Wohlin, 2014). The start set of papers are found by defined search strings and inclusion/exclusion criteria specified in the next section. This work includes both backward and forward snowballing, which are two techniques of the snowballing approach. Detailed steps of each method are described later in this chapter.

Our inclusion (**I**) and exclusion (**E**) criteria are:

- **I.1** The year of publication of works related to MCR is not earlier than 2013 since this year the term and concept of Modern Code Review appeared
- **I.2** The year of other publications that is not related to MCR are not limited
- **I.3** The work is related to modern code review topic
- **I.4** The number of citations is not less than ten since this topic has a narrow scope but at the same time already has many articles
- **I.5** The language in which the work is written in English
- **E.1** The source is outdated
- **E.2** The work is not related to one of the research questions
- **E.3** The article has a few citations
- **E.4** The language of the work is not English

3.3 Quality Check

The crucial part of the systematic literature review is evaluating the found articles by the quality check. The checklist for assessing is presented in Table 5.

3.4 Data Extraction

The data extraction forms must be designed to collect all the information needed to address the review questions and the study quality criteria (Kitchenham, 2004). Table 6 shows the data collection forms and Table 7 contains the final studies included in the final review.

4 PRELIMINARY OF THE RESULTS

After the review of the literature on the topic of non-technical problems, it is possible to note intermediate results on the posed research questions.

Table 5: Quality assessment checks (Ali et al., 2010).

Q1	Is there a rationale for why the study was undertaken?
Q2	Is there an adequate description of the context (e.g. industry,laboratory setting, products used, etc.) in which the research was carried out?
Q3	Has the researcher explained how the study sample (participants or cases) were identified and selected, and what was the justification for such selection?
Q4	Does the study provide description and justification of the data analysis approaches?
Q5	Are limitations of the study discussed explicitly?

Table 6: Data collection forms (Galster et al., 2014) (Fatima et al., 2019).

	Data attribute	**Research Question**
F1	Title	
F2	Author(s)	
F3	Year	
F4	Paper Category (Conference / Report / Workshop / Journal)	
F5	Keywords	
F6	Social and human factors stated	RQ1
F7	Methods for improving MCR	RQ2

RQ1: Social Interactions and the Code Reviews. Several studies have been conducted on how social relationships between members of software teams can influence the review process. It cannot be denied that teamwork always involves social interaction and communication. The software development team is no exception. The process of checking someone else's code is one of the methods of communication when one person communicates with another through the code. As in real life, communication can be different - calm, irritable, funny, etc. But unlike regular communication, the code review carries several consequences, such as fixing the code, skipping critical errors. These factors directly affect the quality of the product. Based on this logic, as well as studies that also showed the relationship between human factors and the quality of the code review, we can conclude that interpersonal relationships affect the code review.

Table 7: List of reviewed studies.

	Title	Author(s)	Year	Category
1	Individual, Social and Personnel Factors Influencing Modern Code Review Process	N. Fatima, S. Nazir, S. Chuprat	2019	Conference
2	Expectations, Outcomes, and Challenges of Modern Code Review	A. Bacchelli, C. Bird	2013	Conference
3	Modern code review: A case study at google	C. Sadowski, E. Söderberg, L. Church, M. Sipko, A. Bacchell	2018	Conference
4	Convergent software peer review practices	P. C. Rigby	2013	Conference
5	Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft	A. Bosu, J. C. Carver, C. Bird, J. Orbeck, C. Chockley	2016	Conference
6	A case study of distributed, asynchronous software inspection	M. Stein, J. Riedl, S. J. Harner, V. Mashayekhi	1997	Conference
7	A preliminary examination of code review processes in open source projects	P. C. Rigby and D. Germán	2005	Journal Article
8	On the shoulders of giants: A new dataset for pull-based development research	X. Zhang, A. Rastogi, Y. Yu	2020	Conference
9	The social structure of open source software development teams	K. Crowston and J. Howison,	2003	Article

RQ2: Prevention of Negative Impact of Non Technical Issues on Modern Code Reviews. The topic of social problems and their impacts are well researched. However, there is a literature gap on preventing them and on possible ways to improve the quality of the review. It follows that this work should be more focused on possible ways to solve social problems in teams.

It seems impossible to avoid interpersonal problems in a process where people are present. But from the first interviews with the developers, we learned that there are still possible options. The first one is to involve several team members in the code review. Another possible solution is to develop a list of criteria by which to check the code inside the team. There are also other options, but we will have to study their effectiveness in more detail in our future work.

5 FUTURE WORK

Qualitative research is an procedure that involves collecting and analyzing the data (e.g., images, sounds, words, and numbers) (Rossman and Rallis, 2003). Such a strategy employs different philosophical assumptions; strategies of inquiry; and methods of data collection, analysis, and interpretation (Creswell, 2009). Its purpose is to learn about some facet of the social world by understanding concepts, opinions, or experiences (Rossman and Rallis, 2003). Qualitative research has different specific approaches such as grounded theory, case study, ethnography, phenomenology, and narrative research. For this work, the grounded theory is most suitable since it helps to study the process of human interaction and generate theories to explain human behavior (Bolderston, 2012).

There are different data collection methods that might be used withing qualitative research approach. One of them is an interview. As was mentioned in the previous chapter, many studies use interviews to learn more about software processes, and in particular code reviews (Bacchelli, 2013) (Bosu et al., 2017) (Rigby, 2013) (Sadowski et al., 2018) (Kononenko et al., 2016). We also decided to survey with professional developers to collect the data regarding the attitude to

Modern Code Review of developers among varying teams and their opinion regarding the objectivity and human factors influencing the review.

The survey consists of several steps, including the preparation phase, execution, and analyzing the results

Survey Design. During the preparation, the interview protocol is set up. A protocol usually includes

- Instructions for the interviewer
- Date, place, interviewer, interviewee
- The questions
- Pilot tests
- A final thank-you statement (Creswell, 2009)

Participant Selection. The target group is developers from the software teams. We suggested to involve participants from the heterogeneous teams so that their work processes may differ from each other. It makes it possible to study the opinion of various categories of developers.

To ensure valid results, is was decided to follow the criteria of one of the previous studies (Bosu 2016). The restriction is to survey developers with sufficient experience. Namely, to interview only those developers who had participated in at least 30 code review requests (Bosu et al., 2017). In connection with the specifics of our research, the study of interpersonal relationships, it was also decided take into account the amount of time during which the survey participant works in the current team. We have set the minimum working time to the six months.

Data Collection. The execution phase requires adherence to the established protocol during the preparation phase. The participants will be asked individually by the established format. The set of questions is the same for all interviewees. But questions may vary according to the semi-structured format to understand and learn more about the participant's opinion.

Data Analysis. Data analysis consists of several consecutive steps such as collecting open-ended data, based on asking general questions, and analysing the information provided by participants (Creswell, 2009).

The results are analyzed in the following order:

- Transcribing interview by organizing and preparing the data for analysis
- Read all the data
- Code the data by classifying the data by words. Coding is the procedure of organizing material into text segments before making sense of information (Rossman and Rallis, 2003).
- Generate a description of the setting or people and categories or themes for analysis by codes
- Represent the description and themes
- Produce qualitative research of the results (Creswell, 2009)

The results of conducted interviews will help us to understand the processes and problems of the code reviewers. Also, we will take into consideration the possible solutions that participants may suggest for their specific team. Depending on the outcomes, it will be possible to conclude the initially set research questions.

6 CONCLUSION

This paper presented the position regarding the non-technical issues in the software code reviewing process. It contains the observations from related researches, the preliminary evaluations, and the proposal. The selected methodology to use is the systematic literature review and the interview. Future work requires data collection and analysis using proposed approaches. Moreover, it would be interesting to explore further the effect of MCR in Open Source (Paulson et al., 2004; Kovács et al., 2004; Petrinja et al., 2010; Fitzgerald et al., 2011; Rossi et al., 2012; Di Bella et al., 2013) and in Agile environments (Maurer et al., 1999; Kivi et al., 2000; Succi et al., 2002; Coman et al., 2014; Janes and Succi, 2014), and when different programming approach are in place, such as mobile (Corral et al., 2011; Corral et al., 2013; Corral et al., 2014) or functional/logic (Marino and Succi, 1989; Valerio et al., 1997; Sillitti et al., 2002; Clark et al., 2004).

ACKNOWLEDGMENTS

We thank Innopolis University for generously funding this research.

REFERENCES

Ali, M. S., Ali Babar, M., Chen, L., and Stol, K.-J. (2010). A systematic review of comparative evidence of aspect-oriented programming. *Information and Software Technology*, 52:871–887.

Bacchelli, A. (2013). Expectations, outcomes, and challenges of modern code review.

Bolderston, A. (2012). Conducting a research interview. *Journal of Medical Imaging and Radiation Sciences*, 43:66–76.

Bosu, A., Carver, J. C., Bird, C., Orbeck, J., and Chockley, C. (2017). Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft. *IEEE Transactions on Software Engineering*, 43(1):56–75.

Brereton, P., Kitchenham, B., Budgen, D., Turner, M., and Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. j syst softw. *Journal of Systems and Software*, 80:571–583.

Clark, J., Clarke, C., De Panfilis, S., Granatella, G., Predonzani, P., Sillitti, A., Succi, G., and Vernazza, T. (2004). Selecting components in large cots repositories. *Journal of Systems and Software*, 73(2):323–331.

Coman, I. D., Robillard, P. N., Sillitti, A., and Succi, G. (2014). Cooperation, collaboration and pair-programming: Field studies on backup behavior. *Journal of Systems and Software*, 91:124–134.

Corral, L., Georgiev, A. B., Sillitti, A., and Succi, G. (2013). A method for characterizing energy consumption in Android smartphones. In *Green and Sustainable Software (GREENS 2013), 2nd International Workshop on*, pages 38–45. IEEE.

Corral, L., Georgiev, A. B., Sillitti, A., and Succi, G. (2014). Can execution time describe accurately the energy consumption of mobile apps? An experiment in Android. In *Proceedings of the 3rd International Workshop on Green and Sustainable Software*, pages 31–37. ACM.

Corral, L., Sillitti, A., and Succi, G. (2015). Software Assurance Practices for Mobile Applications. *Computing*, 97(10):1001–1022.

Corral, L., Sillitti, A., Succi, G., Garibbo, A., and Ramella, P. (2011). Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm. In *Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2011, pages 181–183, New York, NY, USA. ACM.

Creswell, J. (2009). *Research Design: Qualitative, Quantitative, and Mixed-Method Approaches*.

Crowston, K. and Howison, J. (2003). The social structure of open source software development teams.

Di Bella, E., Sillitti, A., and Succi, G. (2013). A multivariate classification of open source developers. *Information Sciences*, 221:72–83.

Fagan, M. E. (1999). Design and code inspections to reduce errors in program development. *IBM Syst. J.*, 38:258–287.

Fatima, N., Nazir, S., and Chuprat, S. (2019). Individual, social and personnel factors influencing modern code review process. In *2019 IEEE Conference on Open Systems (ICOS)*, pages 40–45.

Fitzgerald, B., Kesan, J. P., Russo, B., Shaikh, M., and Succi, G. (2011). *Adopting open source software: A practical guide*. The MIT Press, Cambridge, MA.

Galster, M., Weyns, D., Tofan, D., Michalik, B., and Avgeriou, P. (2014). Variability in software systems—a systematic literature review. *IEEE Transactions on Software Engineering*, 40(3):282–306.

Janes, A. and Succi, G. (2014). *Lean Software Development in Action*. Springer, Heidelberg, Germany.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele Univ.*, 33.

Kivi, J., Haydon, D., Hayes, J., Schneider, R., and Succi, G. (2000). Extreme programming: a university team design experience. In *2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings. Navigating to a New Era (Cat. No.00TH8492)*, volume 2, pages 816–820 vol.2.

Kononenko, O., Baysal, O., and Godfrey, M. (2016). Code review quality: how developers see it. pages 1028–1038.

Kovács, G. L., Drozdik, S., Zuliani, P., and Succi, G. (2004). Open Source Software for the Public Administration. In *Proceedings of the 6th International Workshop on Computer Science and Information Technologies*.

Marino, G. and Succi, G. (1989). Data Structures for Parallel Execution of Functional Languages. In *Proceedings of the Parallel Architectures and Languages Europe, Volume II: Parallel Languages*, PARLE '89, pages 346–356. Springer-Verlag.

Maurer, F., Succi, G., Holz, H., Kötting, B., Goldmann, S., and Dellen, B. (1999). Software Process Support over the Internet. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 642–645. ACM.

Moser, R., Pedrycz, W., and Succi, G. (2008a). A Comparative Analysis of the Efficiency of Change Metrics and Static Code Attributes for Defect Prediction. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE 2008, pages 181–190. ACM.

Moser, R., Pedrycz, W., and Succi, G. (2008b). Analysis of the reliability of a subset of change metrics for defect prediction. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '08, pages 309–311. ACM.

Musílek, P., Pedrycz, W., Sun, N., and Succi, G. (2002). On the Sensitivity of COCOMO II Software Cost Estimation Model. In *Proceedings of the 8th International Symposium on Software Metrics*, METRICS '02, pages 13–20. IEEE Computer Society.

Paulson, J. W., Succi, G., and Eberlein, A. (2004). An empirical study of open-source and closed-source software products. *IEEE Transactions on Software Engineering*, 30(4):246–256.

Pedrycz, W., Russo, B., and Succi, G. (2011). A model of job satisfaction for collaborative development processes. *Journal of Systems and Software*, 84(5):739–752.

Pedrycz, W., Russo, B., and Succi, G. (2012). Knowledge Transfer in System Modeling and Its Realization Through an Optimal Allocation of Information Granularity. *Appl. Soft Comput.*, 12(8):1985–1995.

Pedrycz, W. and Succi, G. (2005). Genetic granular classifiers in modeling software quality. *Journal of Systems and Software*, 76(3):277–285.

Petrinja, E., Sillitti, A., and Succi, G. (2010). Comparing OpenBRR, QSOS, and OMM assessment models. In *Open Source Software: New Horizons - Proceedings of the 6th International IFIP WG 2.13 Conference on Open Source Systems, OSS 2010*, pages 224–238, Notre Dame, IN, USA. Springer, Heidelberg.

Rigby, P. (2013). Convergent software peer review practices.

Rigby, P. C. and Germán, D. (2005). A preliminary examination of code review processes in open source projects.

Ronchetti, M., Succi, G., Pedrycz, W., and Russo, B. (2006). Early estimation of software size in object-oriented environments a case study in a cmm level 3 software firm. *Information Sciences*, 176(5):475–489.

Rossi, B., Russo, B., and Succi, G. (2010). Modelling Failures Occurrences of Open Source Software with Reliability Growth. In *Open Source Software: New Horizons - Proceedings of the 6th International IFIP WG 2.13 Conference on Open Source Systems, OSS 2010*, pages 268–280, Notre Dame, IN, USA. Springer, Heidelberg.

Rossi, B., Russo, B., and Succi, G. (2012). Adoption of free/libre open source software in public organizations: factors of impact. *Information Technology & People*, 25(2):156–187.

Rossman, G. and Rallis, S. (2003). *Learning in the Field: An Introduction to Qualitative Research*.

Sadowski, C., Söderberg, E., Church, L., Sipko, M., and Bacchelli, A. (2018). Modern code review: A case study at google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '18, page 181–190, New York, NY, USA. Association for Computing Machinery.

Scotto, M., Sillitti, A., Succi, G., and Vernazza, T. (2004). A Relational Approach to Software Metrics. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pages 1536–1540. ACM.

Scotto, M., Sillitti, A., Succi, G., and Vernazza, T. (2006). A non-invasive approach to product metrics collection. *Journal of Systems Architecture*, 52(11):668–675.

Siddaway, A., Wood, A., and Hedges, L. (2019). How to do a systematic review: A best practice guide for conducting and reporting narrative reviews, meta-analyses, and meta-syntheses. *Annual Review of Psychology*, 70.

Sillitti, A., Janes, A., Succi, G., and Vernazza, T. (2004). Measures for mobile users: an architecture. *Journal of Systems Architecture*, 50(7):393–405.

Sillitti, A., Succi, G., and Vlasenko, J. (2012). Understanding the Impact of Pair Programming on Developers Attention: A Case Study on a Large Industrial Experimentation. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 1094–1101, Piscataway, NJ, USA. IEEE Press.

Sillitti, A., Vernazza, T., and Succi, G. (2002). Service Oriented Programming: A New Paradigm of Software Reuse. In *Proceedings of the 7th International Conference on Software Reuse*, pages 269–280. Springer Berlin Heidelberg.

Stein, M., Riedl, J., Harner, S. J., and Mashayekhi, V. (1997). A case study of distributed, asynchronous software inspection. *Proceedings of the (19th) International Conference on Software Engineering*, pages 107–117.

Succi, G., Benedicenti, L., and Vernazza, T. (2001a). Analysis of the effects of software reuse on customer satisfaction in an RPG environment. *IEEE Transactions on Software Engineering*, 27(5):473–479.

Succi, G., Paulson, J., and Eberlein, A. (2001b). Preliminary results from an empirical study on the growth of open source and commercial software products. In *EDSER-3 Workshop*, pages 14–15.

Succi, G., Pedrycz, W., Marchesi, M., and Williams, L. (2002). Preliminary analysis of the effects of pair programming on job satisfaction. In *Proceedings of the 3rd International Conference on Extreme Programming (XP)*, pages 212–215.

Valerio, A., Succi, G., and Fenaroli, M. (1997). Domain analysis and framework-based software development. *SIGAPP Appl. Comput. Rev.*, 5(2):4–15.

Vernazza, T., Granatella, G., Succi, G., Benedicenti, L., and Mintchev, M. (2000). Defining Metrics for Software Components. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, volume XI, pages 16–23.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *ACM International Conference Proceeding Series*.

Zhang, X., Rastogi, A., and Yu, Y. (2020). On the shoulders of giants: A new dataset for pull-based development research. In *Proceedings of the 17th International Conference on Mining Software Repositories*, MSR '20, page 543–547, New York, NY, USA. Association for Computing Machinery.