TYPE Original Research
PUBLISHED 11 May 2023
DOI 10.3389/fcomp.2023.1178040

OPEN ACCESS

EDITED BY
Haibin Zhu,
Nipissing University, Canada

REVIEWED BY
Abdallah Qusef,
Princess Sumaya University for Technology, Jordan
Mohamed Wiem Mkaouer,
Rochester Institute of Technology, United States

*CORRESPONDENCE
Artem Kruglov
a.kruglov@innopolis.ru

RECEIVED 02 March 2023
ACCEPTED 17 April 2023
PUBLISHED 11 May 2023

CITATION
Ciancarini P, Kruglov A, Malikova A, Pedrycz W and Succi G (2023) How social interactions can affect Modern Code Review.
Front. Comput. Sci. 5:1178040.
doi: 10.3389/fcomp.2023.1178040

How social interactions can affect Modern Code Review

Paolo Ciancarini[1], Artem Kruglov[2]*, Aygul Malikova[2], Witold Pedrycz[3,4,5] and Giancarlo Succi[1]

[1]Department of Computer Science and Engineering, University of Bologna, Bologna, Italy, [2]Lab of Industrializing Software Production, Faculty of Computer Science and Engineering, Innopolis University, Innopolis, Russia, [3]Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada, [4]Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland, [5]Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Istinye University, Istanbul, Türkiye

Introduction: Modern Code Review (MCR) is a multistage process where developers evaluate source code written by others to enhance the software quality. Despite the numerous studies conducted on the effects of MCR on software quality, the non-technical issues in the MCR process have not been extensively studied. This study aims to investigate the social problems in the MCR process and to find possible ways to prevent them and improve the overall quality of the MCR process.

Methodology: To achieve the research objectives, we applied the grounded theory research shaped by GQM approach to collect data on the attitudes of developers from different teams toward MCR. We conducted interviews with 25 software developers from 13 companies to obtain the information necessary to investigate how social interactions affect the code reviewing process.

Results: Our findings show that interpersonal relationships within the team can have significant consequences on the MCR process. We also received a list of possible strategies to overcome these problems.

Discussion: Our study provides a new perspective on the non-technical issues in the MCR process, which has not been extensively studied before. The findings of this study can help software development teams to address the social problems in the MCR process and improve the overall quality of their software products.

Conclusion: This study provides valuable insights into the non-technical issues in the MCR process and the possible ways to prevent them. The findings of this study can help software development teams to improve the MCR process and the quality of their software products. Future research could explore the effectiveness of the identified strategies in addressing the social problems in the MCR process.

KEYWORDS
Modern Code Review, social interactions, software quality, survey, qualitative analysis

1. Introduction

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; Davila and Nunes, 2021; Malikova and Succi, 2021). The term MCR appeared in 2013 and represents the lightweight version of Fagan inspection. MCR is tool-based and asynchronous process, which distinguishes it from others code review approaches (Bird and Bacchelli, 2013). Asynchrony allows participants to conduct code reviews independently of time and space (Stein et al., 1997). The use of tool-based code review implies adapting a tool to bring structure to the process of reviewing patches and supporting the overall logistics of the process, and there are many such tools that are used in open-source and commercial projects: CodeFlow (used by Microsoft), Gerrit (Google's Chromium and OSS projects), ReviewBoard (VMware), Phabricator (Facebook), and others (Sadowski et al., 2018).

The core of MCR requires at least two people: the author and the reviewer. Some companies involve more than one reviewer, for instance, VMware involves two independent reviewers (Rigby and German, 2006), and Microsoft requires an average of four people (Rigby and Bird, 2013). The process consists of several steps common among different companies: creating, previewing, commenting, addressing feedback, and approving (Sadowski et al., 2018).

One of the most compelling reasons for performing MCR is to prevent developers from inappropriately "protecting" the code they developed, that is, avoiding organizing the process so that no one but them can change their code or even, in the most extreme cases, use it. Moreover, the review provides insight into the code to other developers, promotes the exchange of information among team members, supports them, and improves the overall process and quality of the software (Bird and Bacchelli, 2013). A thorough empirical study conducted at Google evidenced that the main drivers for MCR, from the developers' perspective, are education (teaching or learning), maintenance of organizational standards, and, eventually, prevention of bugs, defects, and other quality issues (Sadowski et al., 2018).

Unlike recent researches that focus on analyzing the artifacts of the online tools used (Ahmed et al., 2017; Asri et al., 2019; Chouchen et al., 2021), our work focuses specifically on a qualitative analysis of the aspects of interpersonal interactions in the MCR process. Consequently, in this work we aim at answering the following research questions:

RQ1: How do social interactions affect the code reviewing process?
RQ2: How to prevent the artifacts induced by social interactions and improve the quality of MCR?

This work is structured as follows. Section 2 presents the state of the art in MCR research. Section 3 describes the selected approaches and the structure of the conducted research. The demographics of the research and data collection process are presented in Section 4. Section 5 shows the results of its implementation. The evaluation and discussion of results in respect to defined research questions as well as overall summary of the work are presented in Section 6, followed by the analysis of validity of the research in Section 7. Finally, we discussed the directions of further research in Section 8 and draw the conclusions in Section 9.

2. Related works

The effects of MCR have been studied in many researches (Ebert et al., 2019, 2021; Nazir et al., 2020; Wang et al., 2021), with experiments conducted at large software companies, such as Microsoft (Bird and Bacchelli, 2013; Rigby and Bird, 2013; Bosu et al., 2017), Google (Rigby and Bird, 2013; Sadowski et al., 2018), Mozilla (Kononenko et al., 2016), and in open-source projects (Rigby and Bird, 2013). These works reveal a significant number of non-technical issues in MCR: distance, review subject, context, customization, and social interaction. Distance can be interpreted as physical or social distance between different teams or different roles. The problem with the subject of the review arises from a lack of comprehension of the code. The problem of context is related to the misunderstanding of the reasons for changing the code. Customization is a problem arising from the specific requirements in each particular company (Uchoa et al., 2020; AlOmar et al., 2022).

A number of researches resulted in development of models and tools for automated evaluation of MCR process and its outcomes (Hijazi et al., 2022; Thongtanunam et al., 2022).

Factors of social interactions include the trust relationship with the author of the code (Zhang et al., 2020), interaction among the MCR participants (history of interactions, frequency, and so on; Bosu et al., 2017; Fatima et al., 2019; Kashiwa et al., 2022), relationships between the team members (Succi et al., 2002; Coman et al., 2014; Bosu et al., 2017), and the perception of the individual author or reviewer (Bosu et al., 2017; Fatima et al., 2019; Kashiwa et al., 2022). Furthermore, individual factors including skills, characteristics, emotions, knowledge and experience, psychological safety, work style, and individual bias, also affect social interactions (Fatima et al., 2019). These kinds of human issues are the area of this research. Specifically, considering how social interactions and individual factors can lead to not objective and misleading reviews.

Problems related to social interaction between developers and affecting the code review process are common to both distributed and co-located teams (Bosu et al., 2017). It is worth considering that interactions become more superficial as the size of the team increases (Crowston and Howison, 2005). Moreover, the researchers found that a few individuals on a team have a large number of interactions, while the rest have only a few (Crowston and Howison, 2005). A number of OSS (Open Source Software) and Microsoft teams were observed to analyze the effects of social factors (Bosu et al., 2017; Alami et al., 2019). The results have shown that concepts such as trust, respect, credibility, and friendship have a significant impact on the code review processes.

The researches mentioned above are mainly focused on the problems of the MCR and their consequences, while the possible solutions to these issues are not well-studied. In this regard, the current research aims not only at investigating social problems described above, we also have a purpose of finding possible ways to prevent them and improve the overall quality of MCR process.

3. Structure of the empirical investigation

This study is based on:

(a) the **Goal-Question-Metric** approach (Basili and Weiss, 1984) for creating a research framework, elaborating the questionnaires, and validating the obtained results, and

(b) the **Grounded theory** research in form of a survey for conducting qualitative research (Bolderston, 2012).

Qualitative research is a procedure that involves collecting and analyzing the data (e.g., images, sounds, words, and numbers; Rossman and Rallis, 2003). Such a strategy uses a different set of philosophical assumptions, research strategies, and methods for collecting, analyzing, and interpreting data (Creswell and Creswell, 2018). Its purpose is to learn about some facet of the social world by understanding concepts, opinions, or experiences (Rossman and

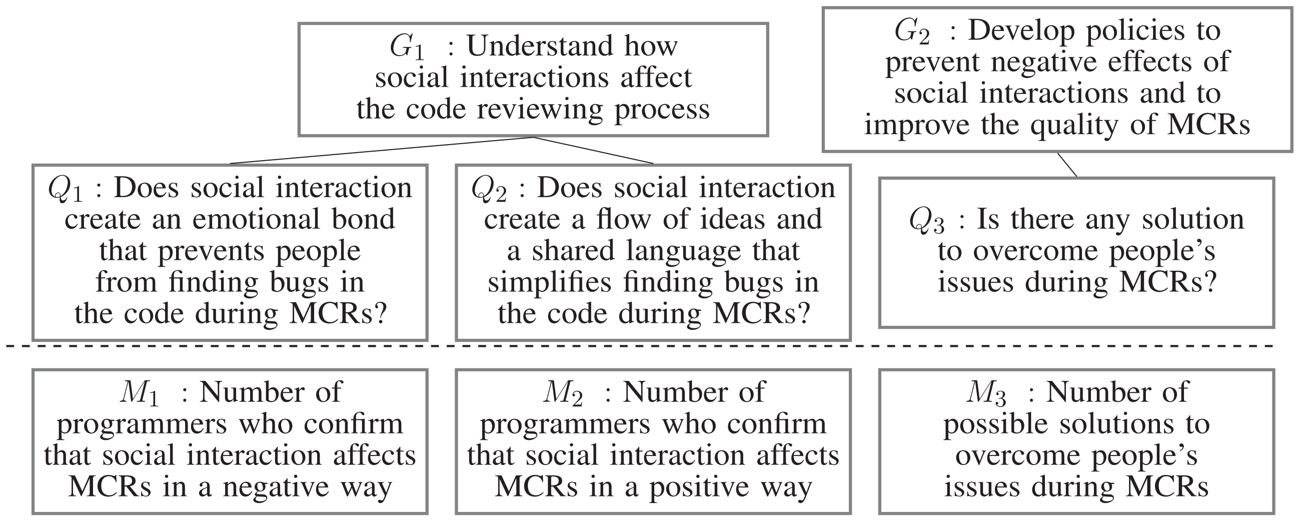
G_1 : Understand how social interactions affect the code reviewing process
G_2 : Develop policies to prevent negative effects of social interactions and to improve the quality of MCRs
Q_1 : Does social interaction create an emotional bond that prevents people from finding bugs in the code during MCRs?
Q_2 : Does social interaction create a flow of ideas and a shared language that simplifies finding bugs in the code during MCRs?
Q_3 : Is there any solution to overcome people's issues during MCRs?
M_1 : Number of programmers who confirm that social interaction affects MCRs in a negative way
M_2 : Number of programmers who confirm that social interaction affects MCRs in a positive way
M_3 : Number of possible solutions to overcome people's issues during MCRs

FIGURE 1
GQM model.

Rallis, 2003). Qualitative research has different approaches such as grounded theory, case study, ethnography, phenomenology, and narrative study (Bolderston, 2012). For our work, the grounded theory is best suited since it helps to study the process of human interactions and generate theories to create theories that explain human behavior (Bolderston, 2012).

3.1. GQM model

In this work we followed the research model suggested by Malikova and Succi (2021): semi-structural interviews with developers of different companies to investigate the topic and gather the statistics. The goals behind conducting the survey are:

1. To understand how social interactions affect the code reviewing process.
2. To define the strategy for preventing the impact of negative social interactions and, thus, improving the quality of MCR.

Using the GQM approach, the defined goals are further elaborated into questions and metrics. The resulted model presented in Figure 1.

3.2. The research strategy

Many studies successfully used interviews to investigate various aspects of the software development process, and the code review in particular (Bird and Bacchelli, 2013; Rigby and Bird, 2013; Kononenko et al., 2016; Bosu et al., 2017; Sadowski et al., 2018; Fregnan et al., 2022). Thus, we also relied on this method to collect data on the attitudes of professional developers toward MCR process and the human factors affecting it.

The survey consists of several steps, including the preparation phase, execution, and analyzing the results (Creswell and Creswell, 2018). Preparation is devoted to the elaboration of the interview protocol, which includes: (a) instructions for the interviewer, (b) date, place, interviewer, interviewee, (c) the questions, (d) pilot tests, and (e) final thank-you statement.

For this survey, we have opted for the face-to-face (individual interview) format. The target group is developers from the software teams. We suggested involving participants from the heterogeneous teams so that their work processes may differ from each other. This allowed us to study the opinions of different categories of developers.

We decided to apply the criteria of the study by Bosu et al. (2017) to ensure the validity of the results. The restriction is to survey developers with sufficient experience—namely, only those who have participated in at least 30 code reviews. Also, since our research is related to the investigation of interpersonal relationships, it was important to take into account the amount of time that the survey participant has been working in the current team. We set the minimum employment criterion of 6 months.

Survey questions were chosen to meet the following criteria: (a) be clear and understandable to interviewees, (b) be related to MCR and its social issues, and (c) not to deal with information that may be protected by a non-disclosure agreement.

The interview script consists of three parts: two questions on demographic, six general questions on the topic (questions 3–8), and nine questions addressing the research objectives (questions 9–14 refer to RQ1 and questions 15–17 refer to RQ2). The complete list of questions for an interview is given in Table 1.

The execution phase required adherence to the established protocol. The interview was conducted with each participant individually. Each interview was accompanied by a sound recorder and a quick-notes tool. The time of interviews varied from 3 to 17 min, depending on the situation. The place depended on the interviewees' preferences as well as their physical location. The set of questions was the same for all interviewees, but the questions could vary according to a semi-structured format in order to understand and learn more about the participant's opinions.

At the end, results were analyzed in the following order:

1. Transcribing interview by organizing and preparing the data for analysis.
2. Read all the data.
3. Code the data by classifying the data by words (Rossman and Rallis, 2003).

TABLE 1 Interview script.

No.	Question	Answer options
1	What is your education level?	• Bachelor's degree • Master's degree • Ph.D. • No university education
2	What is your working experience as a developer?	Open-ended
3	What is your working experience in your current team?	Open-ended
4	How many people work with you in the team?	• 1-4 • 5-9 • 10-15 • 16+
5	What is your work format?	• Remote • Office • Mixed
6	Do you conduct code reviews within your team?	• Yes • No
7	How many hours per week, on average, do you spend reviewing other contributors code?	Opened
8	Why code review is/isn't important?	Opened
9	Has it ever happened that you conducted a code review with a person emotionally close to you?	• Yes, please explain how you felt, and, if possible provide examples • No • I cannot answer
10	Has it ever happened that you conducted a code review with a person that you could not stand?	• Yes, please explain how you felt, and, if possible provide examples • No • I cannot answer
11	Do you think that the identity of the contributor relevant to you when you conduct a code review?	• Yes • No • I cannot answer
12	When you review poorly written code, does it affect your perception of the author of the code?	Open-ended
13	When you review very well written code, does it affect your perception of the author of the code?	Open-ended
14	Could you say that interpersonal relationships affect the objectivity of the code review?	Open-ended
15	If yes, is it a problem in your team?	Open-ended
16	If yes, what possible solutions to these problems do you see?	Open-ended
17	Are there any other problems affecting the code review process within your team?	Open-ended

4. Generate a description of the setting or people and categories or themes for analysis by codes.
5. Represent the description and themes.
6. Produce qualitative analysis of the results (Creswell and Creswell, 2018).

The results of the conducted interviews helped us understand the processes and problems of the code reviewers. In addition, we collected possible solutions that the participants proposed for their particular team.

4. Demographics of the survey

During March and April 2021 we conducted 25 interviews with software developers from 13 different companies. They were carried out according to the established protocol in online and offline formats. To ensure the integrity of the results and to promote the replicability of the research results, we made anonymized audio recordings and processed data[1]

Figure 2 shows the demographic distribution of the participants. Thus, the majority of respondents were male (88%), 12%–female. Most participants (72%) have a bachelor's degree, 12% have a master's degree, 8% are Ph.D. and the rest 8% have no higher education. The working experience of interviewees varies from half a year to 13 years, with the median value is 2.

Information about the teams of participants is shown in Figure 3. According to the respondents, 20% work in an office, 44% work remotely, and 36%—mixed. The number of developers working in the same teams with respondents varies from 2 to 30 people (median is 5). Working time together is from 3 months to 5 years, with the median value of 1 year. And time spending on

1 Available at: https://drive.google.com/drive/folders/17AZoon52nDvVvDK7eUhwczjtP1H9Ec8p?usp=sharing.

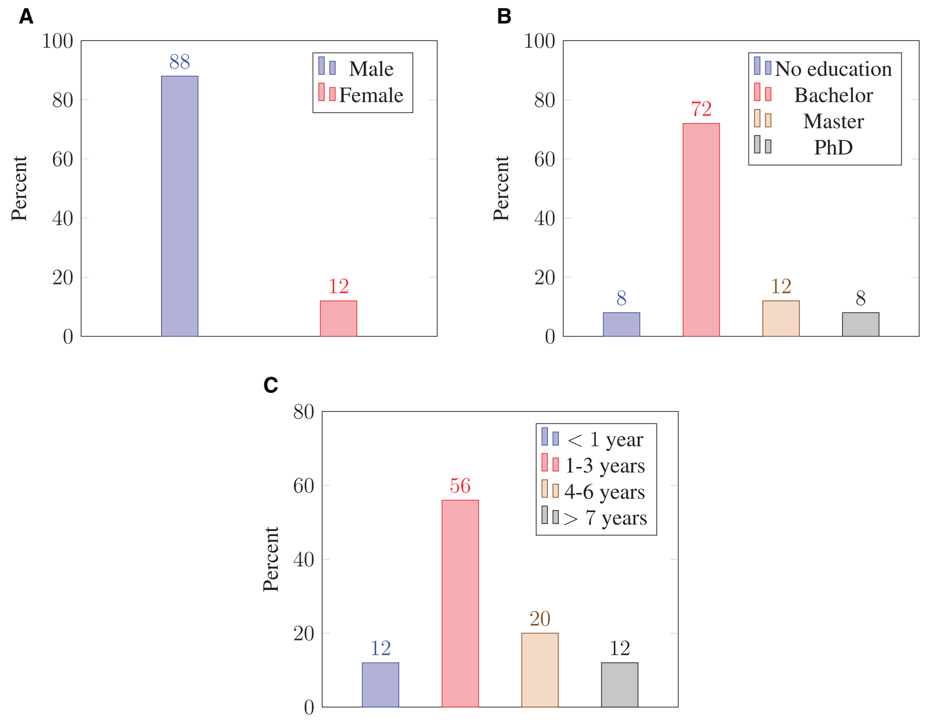

FIGURE 2
Personal information about participants. **(A)** Gender distribution. **(B)** Education level. **(C)** Work experience.

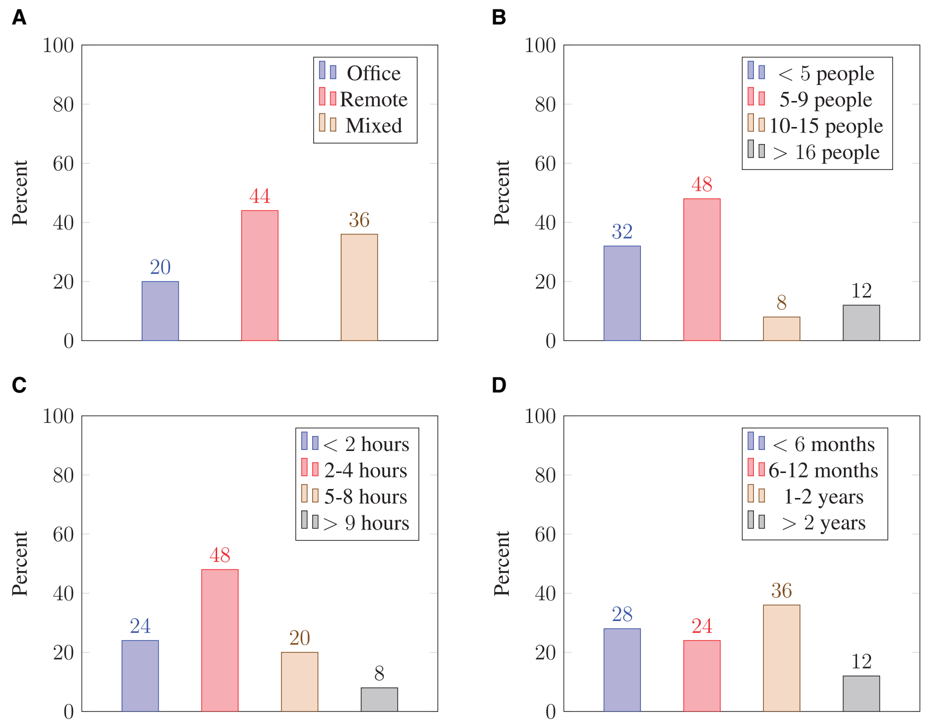

FIGURE 3
Information about working teams. **(A)** Work format. **(B)** Team size. **(C)** Time spending on reviewing other's code per week. **(D)** Time working within the current team.

reviewing other's code per week varies from 30 min to 10 h (median is 2 h).

5. Results of the survey

5.1. Why code review is/isn't important?

One hundred percent of the surveyed participants answered that the code review is an essential part of the development within their team. We also asked developers to mention the reasons why code review is meaningful for them (multiple answers could have been given). And there is a list of all identified reasons for the importance of the procedure, sorted by popularity (Figure 4).

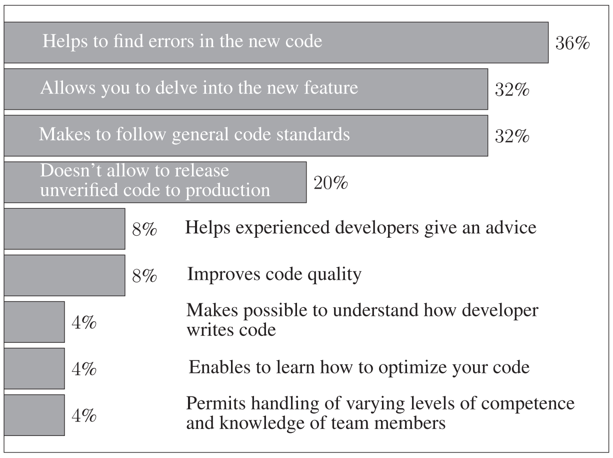

FIGURE 4
Benefits of the code review.

5.2. Has it ever happened that you conducted a code review with a person emotionally close to you? What did you feel?

Most of the participants (68%) answered that they had to check the code of a person emotionally close to them. Opinions about objectivity were divided into two groups. Some developers have said that it's pretty hard for them to be objective when reviewing a person they know well:

> "I had to be not too rude. It was necessary to write something more restrained." (I-1)
>
> "Sometimes I felt admiration." (I-11) "I have to work with my younger cousin. And in this regard, I am stricter with him than with others." (I-14)
>
> "Of course, we look based on our experience, from some of our emotional backgrounds. When we look at a loved one, there's a predisposition, but on the other hand, you can swear him since you're no longer afraid." (I-20)

Others, on the contrary, noted that out-of-work relationships do not matter, and they check everyone identically:

> "There is no difference for me. I conduct reviews for everyone in the same way." (I-10)
>
> "I basically have a neutral collegial attitude toward 98% of people." (I-15)

The second group had significantly more developers (80%) than the first (20%).

5.3. Has it ever happened that you conducted a code review with someone you could not stand? How did that make you feel?

The majority of respondents had never checked the code of a person that they could not stand. Those who did check (28%) provided the following feedback. Most respondents (about 76%) were confident about objectivity:

> "Since we are engaged in a common business, it is better to teach a person who does not know how to do it." (I-20)
>
> "I try to separate the work from the personal, so I was objective." (I-23)

Others (about 24%) felt that their emotions were impacting their judgments:

> "I had an annoyance and desire to find as many errors as possible." (I-1)
>
> "I felt aggression, anger, wish to rewrite his code." (I-4)
>
> "I didn't want to let his code go to production." (I-6)
>
> "If a person was extremely unpleasant to me, I simply ignored his requests." (I-17)

5.4. Do you think that the identity of the contributor is important to you when you conduct a code review?

A significant part of interviewees (75%) answered that identity of the contributor matters when conducting a code review:

> "For me, code review depends on the person's experience. If developer is more experienced, they won't try to delve into his code, and, on the contrary, if I know that developer is a beginner I will be more critical." (I-10)
>
> "This affects the wording of my comment." (I-18)

Others claimed that identity had nothing to do with them when checking the code:

> "If a person has been developing for 3 years, then he cannot make any simple mistakes. Most often, you just need to make sure that everything is fine and I will not thoroughly check any places. But if a person works for a month, then I will check almost every day." (I-16)

5.5. When you review poorly written code, does it affect your perception of the author of the code?

Respondents answered quite differently, but two main points of view can be distinguished. For some of them, the quality of the written code is not related to its author's personality. They do not treat the author of the code as a personality, but as a developer:

> "Rather, I would not make condemnation about this person. I'll just roughly understand how much this person understands the area." (I-11)
>
> "I draw some conclusions regarding his level." (I-21)

While others can draw conclusions about a person by his code:

> "Naturally, if a person is mistaken and repeats, then you can understand what kind of person." (I-2)
>
> "If I know that a person has 5 years of development experience, and at the same time he writes code as an intern, then I will think that the person is not smart enough." (I-25)

5.6. When you review very well written code, does it affect your perception of the author of the code?

The respondents answered this question about the same as the previous one. For the part of interviewers, the quality of the written code is not related to the person's personality:

> "Is there a difference who wrote the code, that is, a rather abstract thing from a professional point of view?" (I-3)

For the second part, the code plays role to form a relationship with the author of the code:

> "If you are not very familiar with a person, then in any case, the attitude toward him will be based on all sorts of little things, including the quality of his code." (I-12)

5.7. Could you say that interpersonal relationships affect the objectivity of the code review?

Most of the developers agreed that relationships at work affect communication processes, and code reviews as well:

> "I usually trust developers with a lot of experience, because I have no doubt that they did something strange." (I-8)
>
> "If he is more experienced, they will try to delve into his code, and if on the contrary, I will be more critical. The personality of the author matters, because for code impulse people I worry more." (I-10)
>
> "When you have confidence in a person, then you start to turn a blind eye at his code." (I-25)

5.8. Is interpersonal relationships a problem in your team?

Few of the participants agreed that interpersonal relationships are a problem in their teams. But some still share the consequences of this problem:

> "Sometimes there are moments that a person finds fault with everything very meticulously." (I-11)

5.9. What possible solutions to non-objective review problem do you see?

In contrast to the importance of code review, the programmers gave only one option for this and the next question. A large number of different options were proposed on how to avoid bias in the code review process (directly corresponds to M_3 of GQM model): (a) involve at least two reviewers in the review process for objectivity, (b) discuss the review criteria with the team, standardization, (c) conversations, (d) reallocation of teams, (e) team extensions, (f) anonymity, (g) help of the team leader, (h) checking soft skills during onboarding, (i) team hierarchy, and (j) enable the creator of the pull request to replace the reviewer.

5.10. Are there any other problems affecting the code review process within your team?

The most popular answers are long-wait for checking pull-request and misunderstanding of the essence of the code. But there are also other aspects: (a) the huge pull-request that hard to review, (b) subjective code vision, (c) assigning the role of the reviewer to a developer who does not know in this area, (d) not using tools for review, (e) communication format, (f) overall involvement, (g) not linking the pull request to the task in the agile board, (h) switching context from tasks to tasks, (i) necessity to quickly release the task and there is no code review in this case, and (j) lack of knowledge of the code base.

6. Discussion

Going back to the GQM model (see Section 3), we made the following observations. With respect to Goal 1, we found two, somehow contradictory, facts:

1. Social interaction seems to create an emotional bond that may prevent people from finding mistakes in code written by peers during MCR sessions (75% of respondents agreed with this statement).

2. Social interaction creates a flow of ideas and a shared language that facilitate finding mistakes in code written by peers during MCR sessions (56%).

With respect to Goal 2, there is a set of possible solutions for overcoming people's issues during MCR provided by interviewees (10 options in Section 5.9).

Regarding the research questions stated for this work, we make the following conclusions.

6.1. How do social interactions affect the code reviewing process? (RQ1)

The influence of interpersonal relationships on code review was proven to be a fact in the conduct and analysis of the interviews. We also found consequences that usually negatively affect the code review process. Among them: missing errors, unwillingness to accept code to production, ignoring, negative emotions that interfere with being objective. It is quite interesting to see an analogy with the results of the MCR anti-patterns analysis conducted by Chouchen et al. (2021)

We analyzed each factor separately, and found out that all of them lead to a deterioration in the quality of the code review. As a result, the quality of the product as a whole may get worse. Missed errors can lead to viral bugs in the product that can affect the user experience (Bird and Bacchelli, 2013). Unwillingness to accept code to production and ignoring requests from the reviewer or the author of the code will affect the teamwork within the team. Lack of communication skills is one of the factors of demotivation and low productivity (Trendowicz and Münch, 2009). Finally, negative emotions not only impede objectivity during the review process (Egelman et al., 2020), but also lead to backslash by the developer, which results in decreased performance.

6.2. How to prevent the artifacts induced by social interactions and to improve the quality of MCR? (RQ2)

Programmers made many suggestions on how to avoid the problems associated with social interactions during the code review. A complete list of proposed options was given in Section 5.9. At the moment, we only have a list of suggested options on ways of improving the review process. Validation of the effectiveness of each of these methods in the context of teams of different sizes and formats requires comprehensive analysis, and is the subject of further research on this topic.

6.3. General conclusions

We have analyzed the results of the survey and come to the following conclusions:

- Code review is an important part of development for any team: with a different format and number of developers.
- Code review is essential for finding bugs, learning a new feature, and maintaining standard.
- The interpersonal relationship between the reviewer and the author of the code can affect the code review process in different ways: missing errors, unwillingness to accept code to production, ignoring, negative emotions that interfere with being objective.
- For most, the personality of the author only matters when taking into account his professional experience.
- The code written by an individual does not affect the perception of his personality, and conclusions are made on his professional qualities.
- There are many other problems in teams related not only to communication between developers. These issues include verification time, the size of the verified code, subjectivity, the level of the verifier, and so on.
- There are several ways to improve the quality of code review according to interviewees' opinions. The most popular are involving multiple reviewers, creation of common criteria, improving soft skills, and others.

7. Validity of the empirical investigation

To be sure of the correctness of the findings, we need to check these findings on validity and reliability (Creswell and Creswell, 2018).

7.1. Internal validity

Threats to internal validity relate to internal factors in our study that could affect our results. To ensure the internal validity of this study, two strategies were employed: participant checking and peer examination. As mentioned earlier, we select only those who have reviewed more than 30 works and have also been working in the current team for more than 6 months. Finally, we conducted the peer examination of the results.

7.2. External validity

Threats to external validity concern the generalization of our findings. The majority of respondents were male and had a high school education, and it can limit the generalizability of the results to other groups of software developers. In addition, although respondents' work experience varied widely, most had relatively limited experience which affect the validity of the study in the context of more experienced developers.

Thus, we should recognize that our findings may not be generalizable to the high extend. Reducing the impact of these factors on external validity may be the direction for further research on this topic.

7.3. Construct validity

To check the comprehensibility and validity of the survey questions, we applied three pilot tests.

The first one was a meeting with a psychology expert during which we rephrase the questions, changed their order, and replaced and added new ones. The need for the consultation with a psychology specialist was that the questionnaire is more social, and the goal is to understand peer impression.

Second, we applied a checklist proposed by Creswell and Creswell (2018) to further improve the question preparation process.

And finally, we performed a test survey with several developers to rephrased some questions, analyze the responses, and perform final improvements to the questions and the process of conducting the interview itself.

8. Future work

This research is just the beginning of studying social issues in the team during the code review. A great deal of work was done in searching for the necessary literature, describing the methodology of the study, and verifying the results. We have received data regarding the questions posed, but we need to investigate them in more detail. In future work, we can study similar researches, which may have already proven the effectiveness of a particular method for improving the code review process, or we can conduct ourfquoteexperiments.

Moreover, it would be interesting to explore further the effect of MCR in Open Source (Paulson et al., 2004; Rossi et al., 2012) and in Agile environments (Coman et al., 2014; Kruglov, 2021), and when different programming approach are in place, such as mobile (Corral et al., 2014) or functional/logic (Clark et al., 2004).

9. Conclusion

MCR has shown to be an effective mechanism to identify bugs in the code (Heumuller, 2021; Han et al., 2022; Hijazi et al., 2022); however, given their intrinsic subjectivity, they can be significantly affected by human factors such as interpersonal relationships, as was shown by Malikova and Succi (2021). In this paper we have investigated impact of social interactions on the code reviewing process. We identified the repercussions of interpersonal relationships that affect the code review process, such as negative emotions and unwillingness to admit the developer's code to production. In 75% cases social interactions create an emotional bond that prevent people from finding mistakes, although in 56% they have positive effect of creating a flow of ideas and a shared language which aids in finding mistakes.

Furthermore, we have collected a number of suggestions on how to improve the quality of MCR process. The most popular opinions are to involve multiple team members in the main review, to develop a list of criteria within the team for which code will be reviewed, and to improve the soft skills of the software development team members.

The results of this work will be useful both to those who involved in software development—software engineers, developers, IT managers—and who want to study this topic further and analyze the human factors in the code review process.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Ethics statement

Ethical review and approval was not required for the study on human participants in accordance with the local legislation and institutional requirements. Written informed consent for participation was not required for this study in accordance with the national legislation and the institutional requirements.

Author contributions

PC, GS, and WP contributed to the conception and design of the study. AK, AM, and GS conducted the literature review. PC and AM created questionnaire and survey design. AM performed the survey. PC, AK, GS, and WP analyzed results of the survey. AK and AM wrote the first draft of the manuscript. All authors contributed to manuscript revision, read, and approved the submitted version.

Funding

This research was funded by Russian Science Foundation, grant 22-21-00494.

Acknowledgments

We thank Innopolis University for generously funding this research.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Ahmed, T., Bosu, A., Iqbal, A., and Rahimi, S. (2017). "SentiCR: a customized sentiment analysis tool for code review interactions," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)* (Hangzhou: IEEE).

Alami, A., Cohn, M. L., and Wasowski, A. (2019). "Why does code review work for open source software communities?," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (Montreal, QC: IEEE). doi: 10.1109/icse.2019.00111

AlOmar, E. A., Chouchen, M., Mkaouer, M. W., and Ouni, A. (2022). "Code review practices for refactoring changes," in *Proceedings of the 19th International Conference on Mining Software Repositories* (Pittsburgh, PA: ACM). doi: 10.1145/3524842.3527932

Asri, I. E., Kerzazi, N., Uddin, G., Khomh, F., and Idrissi, M. J. (2019). An empirical study of sentiments in code reviews. *Inform. Softw. Technol.* 114, 37–54. doi: 10.1016/j.infsof.2019.06.005

Basili, V., and Weiss, D. (1984). A methodology for collecting valid software engineering data. *IEEE Trans. Softw. Eng.* 10, 728–738. doi: 10.1109/TSE.1984.5010301

Bird, C., and Bacchelli, A. (2013). "Expectations, outcomes, and challenges of modern code review," in *Proceedings of the International Conference on Software Engineering* (San Francisco, CA: IEEE).

Bolderston, A. (2012). Conducting a research interview. *J. Med. Imaging Radiat. Sci.* 43, 66–76. doi: 10.1016/j.jmir.2011.12.002

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 Trans. Softw. Eng.* 43, 56–75. doi: 10.1109/TSE.2016.2576451

Chouchen, M., Ouni, A., Kula, R. G., Wang, D., Thongtanunam, P., Mkaouer, M. W., et al. (2021). "Anti-patterns in modern code review: Symptoms and prevalence," in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (Honolulu, HI: IEEE). doi: 10.1109/saner50967.2021.00060

Clark, J., Clarke, C., De Panfilis, S., Granatella, G., Predonzani, P., Sillitti, A., et al. (2004). Selecting components in large cots repositories. *J. Syst. Softw.* 73, 323–331. doi: 10.1016/j.jss.2003.09.019

Coman, I. D., Robillard, P. N., Sillitti, A., and Succi, G. (2014). Cooperation, collaboration and pair-programming: field studies on backup behavior. *J. Syst. Softw.* 91, 124–134. doi: 10.1016/j.jss.2013.12.037

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* (Hyderabad: ACM), 31–37.

Creswell, J., and Creswell, J. (2018). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications

Crowston, K., and Howison, J. (2005). The social structure of free and open source software development. *First Monday* 10. doi: 10.5210/fm.v10i2.1207

Davila, N., and Nunes, I. (2021). A systematic literature review and taxonomy of modern code review. *J. Syst. Softw.* 177:110951. doi: 10.1016/j.jss.2021.110951

Ebert, F., Castor, F., Novielli, N., and Serebrenik, A. (2019). "Confusion in code reviews: reasons, impacts, and coping strategies," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)* (Hangzhou: IEEE). doi: 10.1109/saner.2019.8668024

Ebert, F., Castor, F., Novielli, N., and Serebrenik, A. (2021). An exploratory study on confusion in code reviews. *Empir. Softw. Eng.* 26. doi: 10.1007/s10664-020-09909-5

Egelman, C. D., Murphy-Hill, E., Kammer, E., Hodges, M. M., Green, C., Jaspan, C., et al. (2020). "Predicting developers' negative feelings about code review," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul: ACM). doi: 10.1145/3377811.3380414

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)* (Pulau Pinang: IEEE), 40–45.

Fregnan, E., Petrulio, F., Geronimo, L. D., and Bacchelli, A. (2022). What happens in my code reviews? An investigation on automatically classifying review changes. *Empir. Softw. Eng.* 27. doi: 10.1007/s10664-021-10075-5

Han, X., Tahir, A., Liang, P., Counsell, S., Blincoe, K., Li, B., et al. (2022). Code smells detection via modern code review: a study of the OpenStack and qt communities. *Empir. Softw. Eng.* 27. doi: 10.1007/s10664-022-10178-7

Heumuller, R. (2021). "Learning to boost the efficiency of modern code review," in *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (Madrid: IEEE). doi: 10.1109/icse-companion52605.2021.00126

Hijazi, H., Duraes, J., Couceiro, R., Castelhano, J., Barbosa, R., Medeiros, J., et al. (2022). Quality evaluation of modern code reviews through intelligent biometric program comprehension. *IEEE Trans. Softw. Eng.* 49, 626–645. doi: 10.1109/tse.2022.3158543

Kashiwa, Y., Nishikawa, R., Kamei, Y., Kondo, M., Shihab, E., Sato, R., et al. (2022). An empirical study on self-admitted technical debt in modern code review. *Inform. Softw. Technol.* 146:106855. doi: 10.1016/j.infsof.2022.106855

Kononenko, O., Baysal, O., and Godfrey, M. (2016). "Code review quality: how developers see it," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)* (Austin, TX), 1028–1038. doi: 10.1145/2884781.2884840

Kruglov, A. (2021). "Impact of the communication issues: A case study of IT start-up," in *Frontiers in Software Engineering. ICFSE 2021. Communications in Computer and Information Science*, Vol 1523, eds G. Succi, P. Ciancarini, and A. Kruglov (Cham: Springer). doi: 10.1007/978-3-030-93135-3_8

Malikova, A., and Succi, G. (2021). "Modern code reviews: preliminary results of an analysis of the state of the art with respect to the role played by human factors," in *Proceedings of the 16th International Conference on Software Technologies* (SCITEPRESS - Science and Technology Publications).

Nazir, S., Fatima, N., and Chuprat, S. (2020). Situational factors for modern code review to support software engineers' sustainability. *Int. J. Adv. Comput. Sci. Appl.* 11. doi: 10.14569/ijacsa.2020.0110161

Paulson, J. W., Succi, G., and Eberlein, A. (2004). An empirical study of open-source and closed-source software products. *IEEE Trans. Softw. Eng.* 30, 246–256. doi: 10.1109/TSE.2004.1274044

Rigby, P. C., and Bird, C. (2013). "Convergent contemporary software peer review practices," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013* (Saint Petersburg: ACM Press). doi: 10.1145/2491411.2491444

Rigby, P. C., and German, D. M. (2006). *A Preliminary Examination of Code Review Processes in Open Source Projects*. Available online at: https://flosshub.org/sites/flosshub.org/files/Rigby2006TR.pdf

Rossi, B., Russo, B., and Succi, G. (2012). Adoption of free/libre open source software in public organizations: factors of impact. *Inform. Technol. People* 25, 156–187. doi: 10.1108/09593841211232677

Rossman, G. and Rallis, S. (2003). *Learning in the Field: An Introduction to Qualitative Research, 2nd Edn.* Thousand Oaks, CA: SAGE

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* (New York, NY: Association for Computing Machinery), 181–190. doi: 10.1145/3183519.3183525

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

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)* (Alghero; Sardina), 212–215.

Thongtanunam, P., Pornprasit, C., and Tantithamthavorn, C. (2022). "AutoTransform," in *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, PA: ACM). doi: 10.1145/3510003.3510067

Trendowicz, A., and Münch, J. (2009). Factors influencing software development productivity - state-of-the-art and industrial experiences. *Adv. Comput.* 77, 185–241. doi: 10.1016/S0065-2458(09)01206-6

Uchoa, A., Barbosa, C., Oizumi, W., Blenilio, P., Lima, R., Garcia, A., et al. (2020). "How does modern code review impact software design degradation? an in-depth empirical study," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (Adelaide, SA: IEEE). doi: 10.1109/icsme46990.2020.00055

Wang, D., Wang, Q., Wang, J., and Shi, L. (2021). "Accept or not? An empirical study on analyzing the factors that affect the outcomes of modern code review?," in *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)* (Hainan: IEEE). doi: 10.1109/qrs54544.2021.00104

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* (New York, NY: Association for Computing Machinery), 543–547. doi: 10.1145/3379597.3387489