Bachelor of Science in Software Engineering
Juni 2024

The Impact of Code Reviews on Developer Collaboration and Skill Development

Hamza Al Salkini
Peter Ghaly Bsada

Faculty of Engineering, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Engineering at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Bachelor of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Hamza Al Salkini
E-mail: haal19@student.bth.se

Peter Ghaly Bsada
E-mail: pegh19@student.bth.se

University advisor:
Dr. Ahmad Nauman Ghazi
Department of Department of Software Engineering

Faculty of Engineering
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

This thesis investigates the impact of code review practices on software development, specifically exploring how these practices influence the effectiveness of reviews, enhance collaboration and contribute to developer learning and skill development. Utilizing a combination of surveys and interviews with software developers, we assess several key factors including the experience level of reviewers, the complexity of the code, and the structured feedback mechanisms employed during code reviews. Our findings indicate that experienced reviewers play a critical role in identifying both current and potential future issues, while the complexity of the code can significantly affect the time and effort required for effective reviews. Furthermore, code reviews are shown to substantially improve collaboration and knowledge sharing among team members, facilitated by practices such as pre-review discussions and systematic feedback. Additionally, the quality of feedback received during reviews is important in addressing immediate coding challenges and fostering ongoing professional development. This thesis highlights the need for further research into optimizing code review processes and suggests potential areas where artificial intelligence could be integrated to enhance the efficiency and effectiveness of these reviews.

Keywords: Code Review Practices, Developer Collaboration, Feedback Quality, Software Development and Knowledge Sharing

Acknowledgments

We are grateful to Dr. Ahmad Nauman Ghazi for all of his help and assistance during this study project. We express our gratitude to Blekinge Tekniska Högskola for providing the essential resources and assistance.

Contents

Chapter 1

Introduction

Code reviews are a critical component of the software development process and are implemented by every company engaged in technology. They ensure high-quality code, reducing the number of errors in the final product. Additionally, code reviews offer a vital platform for peer learning and skill development [15].

Code reviews were initially implemented by IBM researcher Michael Fagan in 1974. Since then, they have been used globally for about five decades [19] and they have rapidly grown to be one of the most important software development procedures.

There are several reasons why code reviews have rapidly become one of the most critical practices to implement. Two of the main reasons is debugging and error resolution, which helps minimize code errors and the likelihood of server crashes. This can save large companies significant amounts of money and resources [17].
Another key aspect is the identification and resolution of performance bottlenecks, where experienced developers can provide feedback and suggestions to enhance code speed and performance. Additionally, code reviews offer the benefit of knowledge sharing among colleagues, enabling learning from one another, with senior programmers teaching newer coders.

The code review process is straightforward. Developers are tasked with creating code that performs a specific function. Once completed, the code is submitted for review, where someone checks its adherence to rules and correct implementation. If the code passes the review, it's published, if not, it's returned for corrections. As the article states [18]:

> "Imagine you're writing a book and you've just finished your first draft.
> This draft represents the code that's just been written.
> Your code is like your story.
> It has to be coherent, follow certain rules(e.g., syntax in code and grammar in writing),
> and ultimately achieve its purpose (e.g., to solve a problem or engage an audience).
> The code review process is like the editing phase in writing a book. When you're done writing your draft, you don't immediately publish it. You give it to someone else to read — this person is like the code reviewer."

As it is shown below in Figure 1.1, the code review process begins when a developer completes their code and submits it for review. During the review, peers examine the code for quality, adherence to standards, and potential improvements. Reviewers provide feedback through comments, which the developer addresses by making necessary revisions. Once the code has been updated and re-reviewed, it is approved and then "checked in," meaning it is officially integrated into the main code repository, completing the process. This ensures that only thoroughly reviewed and high-quality code becomes part of the project.

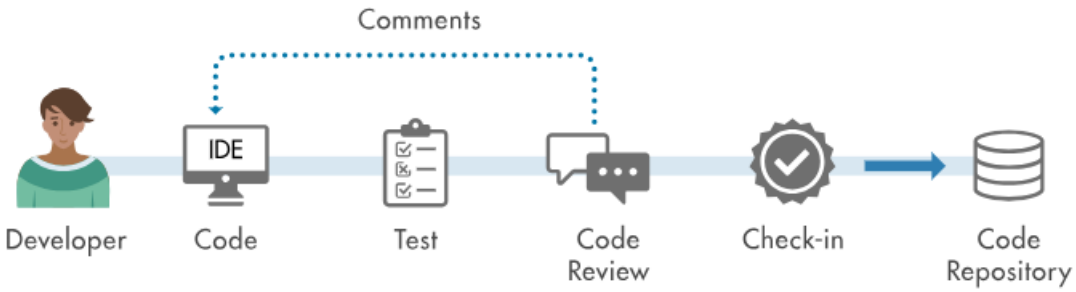

Figure 1.1: Image of the code review process
[14]

However, despite the extensive research on code reviews and their benefits, several gaps remain in understanding the effects of code reviews, particularly in the areas of collaboration, knowledge sharing, and developer skill enhancement. Previous studies have explored various aspects of code reviews, such as their impact on code quality, and the role of non-technical factors. Therefore there is a need for more focused research on investigating the effectiveness of code reviews in encouraging developer collaboration and skill development.

This thesis seeks to address these gaps by investigating the multifaceted impact of code reviews on software development teams. Through a combination of surveys questionnaires and interviews with a variety of software engineers, this study explores three critical aspects: the factors that contribute to the effectiveness of code reviews, the influence of code review practices on collaboration and knowledge sharing, and the role of feedback quality in fostering developer learning and skill development.

Chapter 2

Related Work

In our search for topics that relate to code reviews and developers' collaboration and skills, we found many studies and conference papers that relate to the research we are doing.

One of the conference papers talks about the characteristics of useful code reviews, an empirical study at Microsoft [5]. The study analyzed 1.5 million review comments from five Microsoft projects, uncovering several key factors. For example, it found that the proportion of useful comments made by a reviewer increased significantly during their first year at Microsoft, then stabilized thereafter. Also, the more files are in change the lower the proportion of comments in the code review that will be of value to the author of the code.

In a connected manner, another paper done by Höst and Johansson evaluated the impact of introducing code reviews in a commercial project, using interviews and an experiment to measure effectiveness. Two checklist-based review methods were compared, revealing that code reviews helped identify faults typically found in later stages, though no significant difference was observed between the methods. [12] Despite this, the overall positive results led the organizational unit to continue using code reviews.

In a qualitative study that was done using an exploratory survey of 88 core contributors to the Mozilla project to investigate the code review quality and how developers see it [13]. The findings suggest that the quality review is associated with the thoroughness of the feedback, the familiarity of the reviewers with the code, and the quality of the code itself.

Another study investigated the individual, social, and personal factors influencing the modern code reviews process [10] and found that each of the reported factors can influence the modern code reviews destructively and positively. These results as the authors wrote should be considered while performing modern code review activities to achieve effective results.

A survey on pains and best practices of code reviews [9] has shown that following the practices can ease the process of code review considerably while multiple actionable practices are also in need to support the code review pain. The study summarized 57 practices and 19 code review pains.

A quantitative study presented Knowledge transfer in modern code reviews has shown surprisingly as the authors wrote no significant traces of the knowledge transfer [7]. The authors presented a mining-based study investigating how and whether the code review process helps developers improve their contributions to open-source projects over time. The data used in the study can be relatively important for our research as they have publicly opened it.

In a study called "Four Eyes Are Better Than Two: On the Impact of Code Reviews on Software Quality" the researchers conducted an empirical analysis of three open-source systems to investigate the effectiveness of the code review process on the quality of the code committed submitted. [2] Also, how it can be affected by the number of reviewers and their degree. The study found that reviewed commits are of higher quality than unreviewed ones and are half as likely to contain bugs. It also showed that the code reviewed has significantly higher readability.

Another study called "The Influence of Non-technical Factors on Code Review" focused on how non-technical factors such as personal and organizational aspects influence the code review process outcome. [3] The results of the study showed that the duration of a patch to be reviewed can be influenced remarkably by factors such as the reviewer's workload, the patch writer's experience, the size of the patch, and organizational relationships.

A different research paper called "Code Reviews Do Not Find Bugs. How the Current Code Review Best Practice Slows Us Down" investigates the efficiency of current code review practices. [8] Some factors the study found, are that code review often does not detect critical functionality issues, only 15% of review comments relate to potential defects, the utility of code reviews is strongly dependent on the reviewer's familiarity with the codebase, developers spend around 6 hours per week on reviewing code. The study suggests that reevaluating and refining code review practices should enhance the effectiveness and efficiency of the process.

Within the research titled "Expectations, outcomes, and challenges of modern code review," researchers explore the motivation, challenges, and outcomes of modern tool-based code review. [1] While defect detection is identified as a primary driver, the study found that the actual focus often shifts towards knowledge sharing and improving code readability. Similar to the findings in the previous study "Code Reviews Do Not Find Bugs." A major challenge noted is understanding the changes made, which, much like the issues identified in the other research, affects the effectiveness of the reviews.

2.1 Summary of related work

In summary, previous related works emphasize the crucial role of code reviews in enhancing code quality, identifying potential defects, and improving code readability. While these studies have identified factors that affect code quality, they have not thoroughly explored the factors that contribute to the overall effectiveness of the code

review process. There is also some divergence in opinions regarding knowledge sharing; one study did not observe significant knowledge transfer during code reviews, while another found that the focus has increasingly shifted towards knowledge sharing. Furthermore, none of the previous studies reviewed during this research have addressed the influence of code reviews on developer learning and skill development.

Chapter 3

Method

3.1 Research method

In this research, we have used an empirical approach to explore which factors contribute to the effectiveness of the code review process in software development and how it impacts the developer's collaboration and learning process. We chose survey research as the main research methodology as it is used when the purpose is to collect data from a larger sample of the population and draw generalizable conclusions [11]. The data collection methods used in this thesis are survey questionnaire and interviews. Both qualitative and quantitative data were collected for this thesis.

3.2 Research questions

- RQ1: What factors contribute to the effectiveness of code review processes in software development?
  As the previous studies have identified factors that affect code quality, by this research question we aim to explore what factors contribute to the effectiveness of the code review process, as this is key to ensuring high software quality and efficient development.

- RQ2: How do code review practices impact collaboration and knowledge sharing among development team members?
  By this research question we want to investigate how code review practices impact collaboration and knowledge sharing among developers, as some previous studies had some divergence in opinions regarding knowledge sharing, and they did not talk about how it affects developer's collaboration. which is crucial for better team performance and software outcomes.

- RQ3: What is the role of feedback quality in code reviews, and how does it influence developer learning and skill development?
  This was another gap that we found in the previous studies that no one has examined how feedback from code reviews aids in developer skill growth, as ongoing learning is essential in the fast-evolving field of software development.

3.2.1 Type of data and data collection techniques

In our data collection, we have focused on primary sources specifically for the purposes of this research. These sources provide original, precise information and direct evidence that helps us clarify the research questions. [16].

Both quantitative and qualitative data are required in our research to gather valuable responses that can draw conclusions and produce statistics that highlight the effects of code reviews.

The order of the sample collection started by gathering qualitative data to establish a foundational understanding of the research topic, which involves conducting comprehensive structured interviews with a diverse group of participants. These interviews were designed to dive deeply into personal experiences, perceptions, and detailed insights, which are important for framing the subsequent quantitative phase using a survey.
After analyzing, and identifying themes and patterns in the data gathered from these interviews, we used the analysis to structure the survey. Which focused on the specific details we learned from interviews.

3.2.2 Analysis of data

For summarizing and understanding interview data, we chose to use thematic analysis as the method for analyzing the qualitative data. Thematic analysis is counted as a good approach to briefly and correctly present data, without delving into deeper interpretations. It provides a clear and straightforward overview of the data. [6]

Descriptive analysis is one of the commonly used techniques for analyzing quantitative data. [4] Often it is the first level of data analysis, that helps in summarizing data to identify patterns and provide insights. It has a variation of calculated statics, for example, the percentages, range, and frequency to present a clear picture of the data. However, it does not explain the underlying reasons or relationships as it offers absolute numbers and clear facts about the data.
This technique, including frequency distribution, will be our choice for analyzing the quantitative data.

3.2.3 Validity and Reliability

The validity of this study is primarily limited by the small sample size of 39 survey respondents, which may not fully represent the broader software development community. This limitation suggests that the results should be interpreted with caution.

To address this, we supplemented the survey data with qualitative interviews, providing additional context and depth. Despite this, the small sample size may still affect the reliability of the findings. Future research with a larger and more diverse

sample is recommended to confirm these results.

3.3 Interviews

The interview questions were asked in the same order for all the interviewees, Respondents for the interviews were recruited by reaching out to people we know within the software development industry. We specifically aimed to include individuals with varying years of experience, ensuring a diverse range of perspectives from both junior and senior developers.
We chose to start with warm-up questions to make the interviewer comfortable and to get an overview of what the person does for work and how long they have been working. Once the person answered all the warm-up questions, we began asking the main questions.

After reviewing the literature, we noticed that some important topics were not fully covered: effectiveness, impact on collaboration and knowledge sharing, and developer learning and skill development. We chose these areas because they are important for understanding how code reviews affect software development. We then created our interview questions to focus on these three areas, aiming to fill the gaps we found in the existing research.These main questions are tied to our research questions.
Listing questions below:

3.3.1 Warm-up questions

- Q1: How long have you worked in the Software field?
- Q2: What's your current role?
- Q3: Have you done code reviews in your job before?

3.3.2 Open-ended questions

- Q1: Based on your experience, rank the following factors in terms of most important to least important factors that contribute most to the effectiveness of code review processes in software development.
  Do you think that we are missing some factors? Do you want to add some to the list?
  - Team size
  - Review tooling
  - Reviewer's experience
  - Consistent Standards and Guidelines
  - Team Culture and Communication
  - Knowledge Sharing
  - Quality Feedback

  - Complexity of the codebase

- Q2: How does the experience level or expertise of reviewers affect the outcome of code reviews?
- Q3: How does the complexity of the codebase influence the effectiveness of code reviews?
- Q4: Can you share some examples of which code review practices you follow? For example., regular peer reviews, dedicated review sessions, avoid long code review sessions
- Q5: What code review practices impact collaboration and knowledge sharing directly among members of your development team?
- Q6: In what ways do you think code reviews could be improved to enhance collaboration and knowledge sharing?
- Q7: Can you share an experience where a code review led to significant learning or knowledge sharing within your team?
- Q8: How do code reviews help share common coding standards and best practices within the team?
- Q9: Do you think that code review feedback that you get usually on your code helps you to develop your coding skills? In which way?
- Q10: How does feedback in code reviews align with your personal or professional development goals as a developer?
- Q11: Can you share examples of high-quality feedback you've received or given? What made this feedback particularly effective?
- Q12: What are the most common challenges you face with providing or receiving feedback in code reviews, and how do you overcome them?
- Q13: Do you think the industry would be the same without code reviews?

3.3.3 Interview participant overview

Interviewee	Job title	Experience	Done code reviews before
Interviewee 1	Backend Engineer	3 years	Yes
Interviewee 2	Software developer	3 years	Yes
Interviewee 3	Frontend developer	4 years	Yes
Interviewee 4	Software developer	3 years	Yes
Interviewee 5	Software developer	7 months	Yes
Interviewee 6	Full-Stack Developer	4 years	Yes
Interviewee 7	Software developer	1.5 years	Yes
Interviewee 8	Frontend Developer	1.5 years	Yes
Interviewee 9	Software Engineer	4 years	Yes
Interviewee 10	Full-Stack Developer	1 year	Yes

Table 3.1: Interviewees information.

3.4 Survey

In the survey, we received 39 responses from a diverse range of roles within the software development field, encompassing various years of experience. We shared the survey link with people we know and encouraged them to also share it with others in the software development field. This approach helped us reach a wider audience within the industry In the next tables, we grouped the participants based on years of experience, role and team size.

3.4.1 Survey questions

1. How many years of experience do you have in Software Development?
2. What role do you mainly work in?
3. What is the size of the team you are working with?
4. How does the experience level or expertise of reviewers affect the outcome of code reviews?
5. How does the Team Culture and Communication affect the collaboration in code reviews?
6. Do you agree that Consistent Standards and Guidelines contribute to the effectiveness of code review processes?
7. Is feedback during code reviews provided in a timely manner?
8. How does Pre-review discussions impact the knowledge sharing between team members?
9. Do you agree that teaching the culture of taking criticism and learning from it is important to enhance the collaboration and knowledge sharing?
10. Do you agree that code review feedbacks that you get usually on your code helps you to develop your coding skills?
11. Do you agree that the industry of Software Development would not be the same without code reviews?

After analyzing the responses from the interview questions, we identified areas where the results were not strong or clear enough. To strengthen and complement these findings, we developed the survey questions to gain additional insights and confirm the themes that emerged from the interviews.

3.4.2 Survey participant overview

Number of participants	Range of years of experience
1	less than 1 year
21	1-3
9	4-6
8	7-10

Table 3.2: Participants years of experience.

Number of participants	Role
1	Scrum master
14	Backend developer
7	Frontend developer
15	Fullstack developer
2	Tester

Table 3.3: Participants role.

Number of participants	Role
2	1-2
18	3-5
19	6-11

Table 3.4: Participants team size.

Chapter 4

Results and Analysis

4.1 Thematic map based on interviews

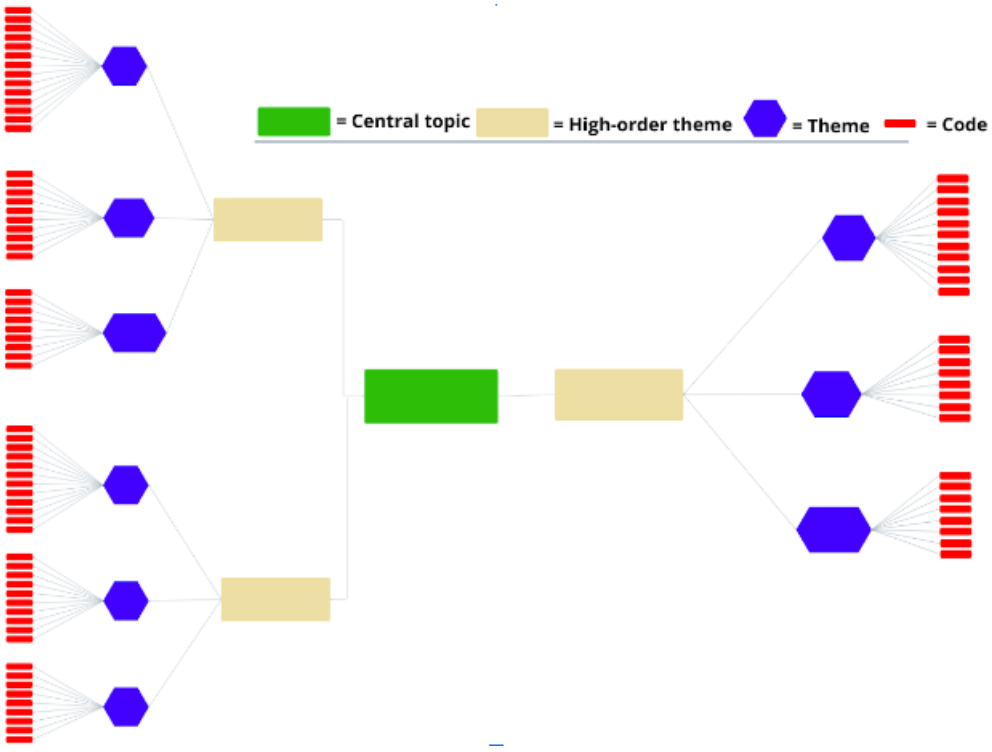

Figure 4.1: Structure of thematic map

4.1.1 Effectiveness

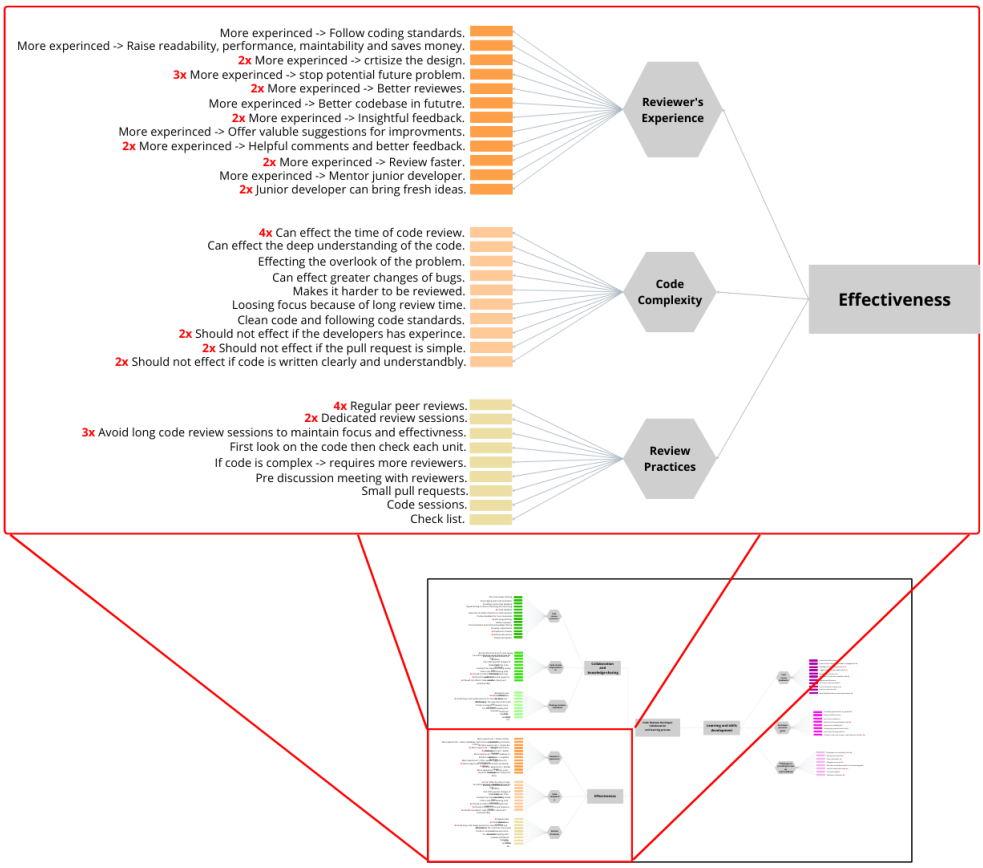

Figure 4.2: High-order codes and theme for Effectiveness

In the context of software development, the effectiveness of code reviews is pivotal for maintaining high standards of code quality and ensuring that software functions as intended. To understand and enhance the effectiveness of these reviews, it is essential to analyze specific factors that play a crucial role. This section categorizes these influential factors into three key areas: Reviewer's Experience, Code Complexity, and Review Practices. Each of these factors contributes uniquely to the success of code reviews by impacting the depth, efficiency, and outcome of the review process.

4.1.1.1 Reviewer's Experience

Code reviews do not have formal requirements regarding the experience level of the reviewer. Whether one is a senior developer or a junior developer, everyone is expected to engage in code reviews. However,by analysing the answers from our interviewees, our findings indicate that the depth of experience significantly impacts the quality of the code review process.

"The experience level of reviewers greatly **influences the outcome of code reviews**. More experienced reviewers tend to provide **more insightful feedback**, **catch potential issues earlier**, and offer **valuable suggestions for improvement**." - Interviewee 2
Number of interviewee that had the same opinion: 8x

"*The feedback is also better when the reviewer has more experience.*" - Interviewee 10
Number of interviewee that had the same opinion: **5x**

By analyzing all the answers from our interviewees, we can see that the majority shared the same opinion that experienced reviewers can affect code reviews. In the quotes from one interviewee, the person covered what 80% of the interviewees said. He mentioned four factors that were repeated by other participants with the same opinion. Experienced reviewers significantly influence the outcomes of code reviews and can also catch potential issues earlier because of the experience he has in the software field. Among the respondents, 40% of the 80% specifically stated that the more experienced the reviewer, the better the feedback and suggestions for improvement.

For instance, interviewee 6 highlighted that experienced reviewers not only find more issues but also review faster, mentor junior developers, grasp the project context better, and enforce coding standards.

In conclusion, it is evident that the experience of the reviewer greatly affects the quality of the code review. This underscores the importance of involving experienced developers in the code review process to enhance the effectiveness and efficiency of software development.

"I do not think that it affects too much." - Interviewee 7
Number of interviewee that had the same opinion: 2x

While the experience of reviewers typically enhances the effectiveness of code reviews, insights from two participants highlighted that it doesn't affect too much. They didn't explain why it didn't affect much, but they believed it didn't have a significant impact.

Interviewee 4 added, "It does not affect a lot. Junior developers can also bring fresh ideas to the team." These insights emphasize that junior developers, despite their lack of extensive experience, can introduce fresh ideas and perspectives. These individuals are less likely to be constrained by conventional practices, making them more open to exploring innovative solutions.

4.1.1.2 Code Complexity

Projects in any organization can vary greatly in size and complexity. When we looked at our data, it was hard to definitively say how much the complexity of a project affects code reviews. People have different opinions on this, some think that more complex projects make code reviews more difficult and time-consuming, while others don't see a clear link.

"The complexity of the codebase can affect the effectiveness of code reviews by influencing the time required for review, the depth of understanding needed, and the likelihood of overlooking subtle issues."
- Interviewee 2

"Complex code makes reviews harder, slows them down, and requires expert reviewers for effective feedback."
- Interviewee 6

"The more complex the code is the more it costs time and effort to do an effective code review. Some examples of complexity causes are not using the correct code standards, design patterns and the object oriented programming effects the time of review."
- Interviewee 9

As the complexity of the codebase increases, the challenges associated with conducting thorough and effective reviews also rise. Complex code requires more time for review, and a deeper understanding of the material, and increases the chances that subtle issues may be missed. This complexity often stems from not adhering to standard coding practices, such as the consistent application of design patterns and principles of object-oriented programming. Consequently, complex code not only demands more effort and time to review but also necessitates the involvement of expert reviewers to provide meaningful feedback and ensure the code's integrity.

"This should not affect the code review effectiveness a lot, because good programmers should be able to release PRs that are dedicated to only one thing, and keep it simple as possible so the single PR is simple regardless the whole code base is complex"
- Interviewee 1

"The codebase should not affect the effectiveness, the developers should instead change the amount of code being reviewed to scale down large or complex applications"

- Interviewee 3

"It's not about the size of the codebase, but rather about the readability and understandability of the submitted code. If the code is written in a clear and understandable manner, developers can expect quicker reviews."
- Interviewee 5

Several interviewees believe that the complexity or size of a codebase shouldn't make code reviews less effective if developers follow good practices. Interviewee 1 suggests that skilled programmers should aim to keep their pull requests (PRs) focused on just one thing and as simple as possible. This way, each PR is easy to handle, no matter how complex the overall project might be. Interviewee 3 advises that developers should adjust how much code they review at once, especially for larger or more complicated parts of the project. Interviewee 5 adds that what matters is how clear and easy to understand the submitted code is. If the code is well-written and straightforward, it will lead to faster and more effective reviews.

Because of these mixed views, more research is needed to truly understand how project complexity might impact code review processes in various projects.

4.1.1.3 Review Practices

Understanding the importance of structured review practices in maintaining efficiency and clarity, we asked interviewees about the specific practices they follow during code reviews. Here's what one of them shared:

"A good practice we have in the team is that the person who opened a PR (if the PR is long and complex) goes through the problem and the solution with the potential reviewers before the actual review process, this creates a better understanding and results in better reviews. Another good practice is to have dedicated personal time slots for doing code reviews, this prepares the mind better for the process and not jump to the code between meetings while the mind is not at all prepared."
- Interviewee 1

Summary of all answers:

- Regular peer reviews. 4x
- Dedicated review sessions. 2x
- Avoid long code review sessions to maintain focus and effectiveness. 3x
- First look at the code then check each unit.
- If the code is complex -> requires more reviewers.
- Pre-discussion meeting with reviewers.
- Small pull requests.
- Code sessions.
- Checklist.

This response highlights the practical steps taken to ensure that the review process is as effective and organized as possible, showcasing how tailored practices can directly impact the quality and speed of code reviews.

4.1.2 Collaboration and knowledge sharing

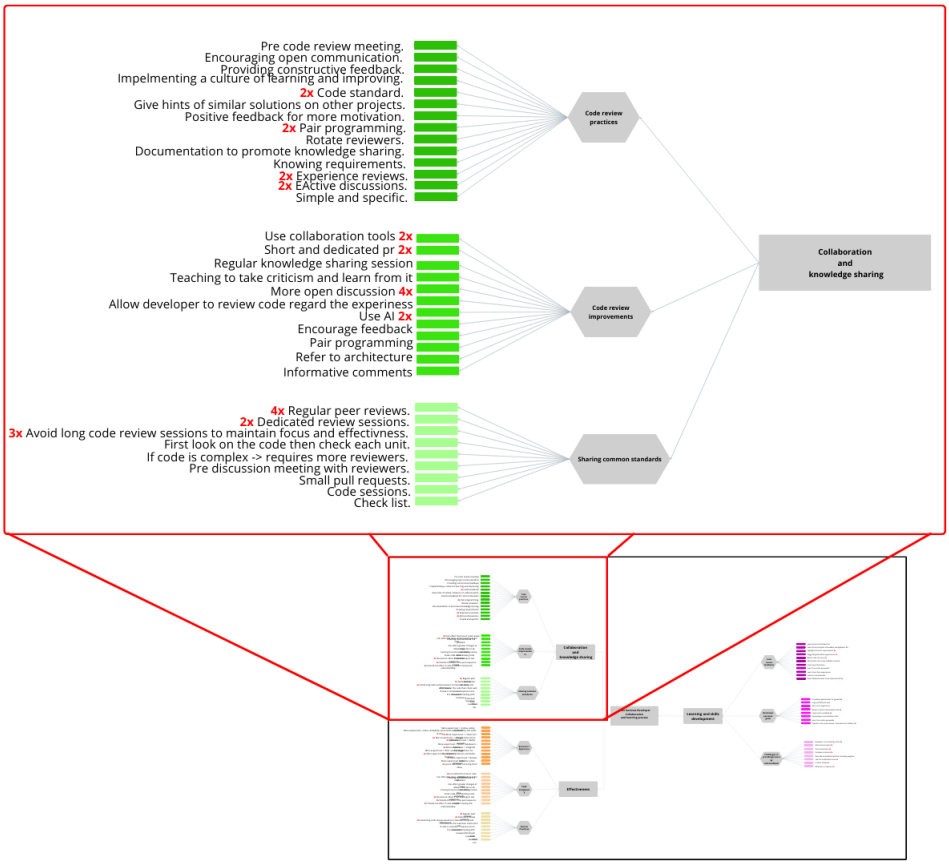

Figure 4.3: High-order codes and theme for Collaboration and knowledge sharing

Code reviews play a critical role in enhancing collaboration and facilitating knowledge sharing among software development teams. To understand this impact more deeply, we have categorized the responses from our interviews into three main themes. These themes encapsulate the various ways in which code review practices contribute to team dynamics and information exchange.

4.1.2.1 Code review practices

This theme explores how various code review practices significantly influence team collaboration and the sharing of knowledge. By analyzing comments from collected interviewees, we can identify which factors most effectively promote collaboration and enhance knowledge sharing. Key practices highlighted by participants include the following: code standards, pair programming, experience reviews, and active discussions.

To illustrate the insights from our interviewees, we selected the quotes that most strongly contribute to this theme. For instance, one interviewee describes an effective practice within their team:

"A good practice we have in the team is that the person who opens a PR, especially if the PR is long and complex, goes through the problem and the solution with potential reviewers before the actual review process. This allows the whole team to look at the problem together, discuss it, and then proceed to individual reviews, ensuring that the team moves in one direction, rather than everyone going in separate directions."
- Interviewee 1

This highlight the importance of pre-review meetings, particularly for complex pull requests. By discussing the problem and proposed solutions collectively before initiating formal code reviews, the team ensures a unified understanding and strategy. This practice not only streamlines the review process but also fosters collaboration and consensus-building among team members, significantly reducing the likelihood of divergent or conflicting feedback during the review phase. Such proactive communication helps align the team's efforts and enhances the efficiency of the development process.

Another interviewee highlights the role of experienced reviewers:

"It's about sharing the knowledge of the code reviewer, especially with experienced reviewers. Discussions on complex code together with reviewers can share viewpoints that can help develop the knowledge of the developers."
- Interviewee 8

This insight points to the importance of 'experience reviews,' where experience code reviewers are involved. Having experienced reviewers on board not only improves the quality of the code review process but also serves as an essential educational tool for the team. Through detailed discussions on complex aspects of code, less experienced developers gain valuable insights and learn advanced coding techniques. Such interactions are instrumental in building a knowledgeable and skilled team, ensuring that expertise is effectively passed down and shared among all members. This practice not only enhances individual developer skills but also strengthens the team's overall

capability to handle challenging programming tasks.

4.1.2.2 Code review process improvements

This theme contains ideas on how the industry can improve, to enhance collaboration and knowledge sharing. By identifying the data we have on how to improve code review collaboration and knowledge sharing can we then identify the flows that the industry has. The majority of interviewees said short and dedicated pull requests, collaboration tools, use AI. Something that came up a lot was a more open discussion.

"*By making the PRs as short and dedicated to one thing as possible, not too many problems should be solved in 1 PR.*" - Interviewee 1
Number of interviewee that had the same opinion: 2x

By analyzing the answers we had 2 people said that pull requests should contain a small amount of code to make the changes easier to understand. The reviewer's comments should be informative, providing tips with links when possible. If a comment becomes too long or is difficult to explain in a few lines, the reviewer should schedule time with the developer to provide a more detailed explanation.

"*Encourage feedback, use collaboration tools, pair programming, knowledge sharing sessions, mentoring programs*". - Interviewee 6
Number of interviewee that had the same opinion: 5x

50% of the interviewee had the same opinion and said to enhance collaboration and knowledge sharing through code reviews, companies should conduct regular knowledge-sharing sessions and use collaborative tools to facilitate discussions and document insights. Encouraging feedback through collaboration tools, pair programming, and mentoring programs can significantly improve the process. Additionally, having one-on-one sessions with developers who have submitted code for review can help them understand and correct mistakes. While leaving comments is beneficial, adding these one-on-one sessions would be even more effective, helping new developers to drastically improve their skills and coding practices.

"*Right now we have AI so I will think using AI to pinpoint more simple stuff, that would decrease the effort that is needed from the reviewer letter as a normal human that is going to review it. Say if I forget a print state, the AI will see it directly, that would decrease the effort that is needed from a normal person.*" - Interviewee 10
Number of interviewee that had the same opinion: 2x

Artificial intelligence (AI) is starting to be used more in code reviews, although it's still a new idea. Interviewee 10 had a big interest in AI and he said that AI can automatically find simple mistakes in code, like missing print statements. This makes the review process easier because it means human reviewers don't have to check everything themselves. They can focus on more complicated issues that AI might not catch, it is also a chance that a normal person can't catch the problem but when AI is so new and companies haven't implemented it, they put more trust

in a developer. By using AI, teams can work more efficiently because they spend less time on easy fixes and more time on important problems.

Implementing these strategic practices ensures that code reviews are more than just a quality assurance measure. They become essential learning and development experiences that strengthen team dynamics and improve knowledge dissemination.

4.1.2.3 Sharing common standards

Every technology company that conducts code reviews typically adheres to certain coding standards. These standards are crucial because they help developers write clean and efficient code, which in turn leads to software with fewer bugs. Following a common set of standards ensures that the code is consistent and easy to understand across the team, which is essential for maintaining quality.

An interviewee shared examples of the coding standards their team follows:

"We adhere to the coding standards that we have agreed upon as a team. For instance, we ensure proper naming of commits, include task reference numbers from JIRA in our code commits and maintain consistent variable naming and placement."
- Interviewee 8

These practices are not just about keeping the code tidy, they also make the review process more straightforward and efficient. When everyone follows the same rules, it's easier for reviewers to spot issues and for developers to incorporate feedback without confusion. This collaborative effort helps in building robust and reliable programs.

As this developer mentioned, it is significantly important to establish coding standards early:

"I think one of the most important things is to set those standards before starting the development as those standards will be used in the code reviews."
- Interviewee 10

This emphasizes how crucial it is to have common standards in place from the outset to guide effective code reviews.

4.1.2.4 Interviewee shared experience with code review

We asked the interviewees to share experiences where a code review led to significant learning. By asking this question we can identify which factors contribute to skill development. Here are some of their responses:

"*During 1 of the biggest releases we had, the senior engineer got to explain the design they implemented to junior engineers which was very useful, and made the process of reviewing much simpler for us.*" - Interviewee 1

"*A memorable experience where a code review led to significant learning within my team was when a senior developer provided detailed feedback on my code structure, which helped me understand better architectural principles and design patterns.*" - Interviewee 2

By analyzing the answer we notice that everybody talked about the feedback they received and how it helped them. 90% of the interview experiences where a code review led to significant learning was when somebody gave them feedback to improve. For example we have Interviewee 4 say, "A coworker told me that .map in React is almost twice as fast as .forEach. I avoid forEach in almost all cases now." Feedback is a big part of the learning skills.

4.1.3 Learning and skills development

Figure 4.4: High-order codes and theme for Learning and skill development

Code reviews are a crucial part of keeping clean, error-free code as well as supporting developers' ongoing education and skill development. Although finding and fixing bugs and errors is frequently the main goal of code reviews, their importance goes far beyond these technical details. Code reviews are an essential teaching tool that gives developers the chance to improve their coding skills, pick up new tricks, and deepen their knowledge of the codebase. This procedure not only raises the software's quality right away but also helps developers advance professionally over time, keeping them competent and informed in a constantly changing technical environment.
Three main themes were identified:

4.1.3.1 Code review feedback's

Receiving feedback from code reviews is a big help in becoming a better developer. However, how this input is received has a big impact on how beneficial it is. Even helpful and correct criticism loses impact if it is not met with an open mind and willingness to grow. Because of this, developers must respond positively to code review input and are prepared to make the necessary modifications in response to the

recommendations made. This method improves the codebase's overall quality while simultaneously fostering individual growth.

"Feedback received in code reviews helps develop coding skills by highlighting areas for improvement, suggesting alternative approaches, and exposing developers to diverse perspectives and best practices."
- Interviewee 2

"It helps me think about how we are structuring our code currently and what we can do to improve it. It shows me where i usually make mistakes so i can improve on that and double check the typical feedback places"
- Interviewee 3

"Yes, code review feedback helps improve coding skills by highlighting errors and suggesting improvements enhances understanding and prevents repeating mistakes, and by learning from experienced reviewers exposes me to better coding practices and techniques."
- Interviewee 6

"It helps 100%, It learns communication, accepting other perspectives, learn how to prepare proofs that supports my opinion to discuss it with others, know how others thinks and learn from their experience."
- Interviewee 8

"How good the feedback on the code review determines the amount of knowledge that developer gains in this process. I think I have learned a lot and still learning more with each developed code."
- Interviewee 9

"Yes it helps, every day I learn something new."
- Interviewee 10

Feedback from code reviews is highly valued by developers for its role in enhancing their coding skills and professional development. This feedback helps them identify where they need improvement, introduces them to new ways of coding, and exposes them to best practices from more experienced colleagues. It also helps developers recognize their common errors and encourages them to think more critically about how they structure their code.

Beyond just improving coding skills, feedback in code reviews teaches developers important soft skills like effective communication, openness to other perspectives, and how to present and discuss their ideas convincingly. As developers regularly receive and apply this feedback, they continue to learn and grow with every project, constantly expanding their skills and knowledge.

4.1.3.2 Developer personal goals

When asked how feedback from code reviews supports their personal and professional development goals, the majority of interviewees responded positively. They emphasized that code reviews are crucial for continuous learning and avoiding stagnation in their technical and professional skills. This process ensures that developers do not remain confined to the same level of knowledge but instead are encouraged to expand their expertise and tackle new challenges. According to the feedback, code reviews effectively help developers meet their growth objectives, playing a significant role in their ongoing development and career progression.

"Feedback in code reviews aligns with my professional development goals as a developer by providing opportunities for growth, learning, and continuous improvement."

- Interviewee 2

"Getting feedback always feels good since I can improve in different areas. Without the feedback it is hard to see what could be wrong with the code I'm writing."

- Interviewee 3

"Feedback is very valuable for me and through feedback one does grow it doesn't matter if it's positive feedback or constructive feedback. I have grown a lot in the past 7 months. Now I can see the bigger picture which helps me to write maintainable code and is easy for others to understand even if they have not worked in the same repo as me."

- Interviewee 5

" I think it helps in developing the communication skills between the team members, where developers get to discuss the code and show thier different perspectives about the code. This helps for example the new developers getting to know and learn more about the code. One of the important things is to understand the feedback

and why it was given as it will help developing the skills in the future."
- Interviewee 9

Everyday I improve my skills because of the code review we have.
- Interviewee 10

Each response underscores the value of feedback, not just in refining technical skills but also in facilitating a deeper engagement with the team and the overall project work.

4.1.3.3 Challenges in providing/receiving code feedback

While code reviews are invaluable for ensuring high-quality software and fostering developer growth, they are not without their challenges. The complexity and demands of code review can sometimes make it a difficult process to manage effectively. Various factors contribute to these challenges, impacting the ease and efficiency with which reviews are conducted.

Here are some perspectives from interviewees on what they find challenging about code reviews:

"One of the most common challenges is by receiving feedback based solely on the reviewer's opinion and not achieving any real value, it can be overcome by arguments and by getting a more senior person to judge between the feedback and the original code."
- Interviewee 1

One common challenge in code reviews is getting feedback that feels more like a personal opinion than helpful advice. Interviewee 1 notes that this kind of feedback can lead to arguments instead of improvements. To solve this, they recommend asking a more experienced team member to step in and decide if the feedback is useful or not. This helps ensure that the feedback is truly valuable and can help improve the code.

"The hardest challenge for me is when I review a code which is hard to understand it takes me longer to leave a review which can affect the ci/cd cycle. When this happens I usually ask another member for help."
- Interviewee 5

Interviewee 5 describes a common challenge in code reviews: struggling with code that is hard to understand. This complexity can slow down the review process, po-

tentially delaying the continuous integration/continuous delivery (CI/CD) cycle. To manage this issue, Interviewee 5 often seeks help from another team member. This collaboration helps clarify confusing aspects and speeds up the review process, ensuring that it doesn't hold up the overall project timeline.

"Common challenges: defensiveness, time constraints, communication, knowledge disparity. I usually overcome them by: emphasizing improvement, setting realistic expectations, clear communication, fostering a learning culture."
- Interviewee 6

Interviewee 6 identifies several typical challenges in code reviews: defensiveness from developers receiving feedback, time constraints that pressure the review process, communication issues, and disparities in team knowledge levels. To address these challenges, Interviewee 6 employs specific strategies: they focus on improvement rather than criticism, set realistic expectations to manage time better, ensure clear communication to avoid misunderstandings and foster a learning environment that helps bridge knowledge gaps among team members.

A consistent challenge noted by some of the interviewees is the difficulty some developers face in accepting criticism or differing opinions. This issue complicates the code review process, as the primary role of a reviewer is to evaluate and suggest improvements to the code. When developers resist feedback, it not only hinders their own growth but also makes it challenging for reviewers to perform their duties effectively. Addressing this issue requires fostering a team culture where feedback is seen as a valuable tool for development rather than a personal critique.

4.1.3.4 Would the industry be the same without code reviews

We asked people whether the industry would be the same without code reviews. From their responses, we can determine the importance of code reviews.

"Definitely not, code reviews are not only for spotting errors, but also for improving the design and overall architecture, if code reviews do not exist, then we will have more bad codebases that are very hard to deal with and more error prone code." - Interviewee 1

"The industry would not be the same without code reviews. Code reviews play a crucial role in ensuring software quality, fostering collaboration and learning, and maintaining codebase consistency and maintainability. They are integral to the software development process and contribute significantly to overall product quality and team effectiveness." - Interviewee 2

"Definitely no, without code reviews it is hard to get high quality of code. Writing code is easy but maintaining it is the hard part. Therefor clean code with good readability helps a lot in the lifetime of the software." - Interviewee 8

"No, Code review is a big part of the industry and important." - Interviewee 10

This is a summary of the responses we received. Everyone agreed that the industry would not be the same without code reviews. We can now see that code reviews are crucial for both knowledge sharing and the quality of the software.

4.2 Descriptive analysis

4.2.1 Survey

In this section, we will present a descriptive analysis on the data that we collected through the questionnaire. The survey questions were based on key insights from the interviews. The interviews highlighted important themes and challenges in code reviews, which were then turned into survey questions to see how common these issues were among a larger group of developers. This way, the survey directly reflected real experiences and focused on the most relevant aspects, like effectiveness, collaboration, knowledge sharing, and how code reviews help developers learn. To conduct the survey, we designed the questions based on the analysis of the interview responses, ensuring that the survey could provide meaningful results grounded in the initial qualitative findings. We have received 39 responses from different roles in the software development industry.
Below are graphical statistics that show answers grouped by each research question subject.

4.2.2 Effectiveness

The data presented in the coming graph provides insights into the effectiveness of code reviews and explores whether various factors influence the review process. By analyzing this data, we aim to determine the extent to which these factors affect the outcomes of code reviews.

In Figure 4.5, 4.6 and 4.7 we examine whether specific factors impact the outcomes of code reviews. By analyzing the data presented in the graph, we aim to define the area in which these factors influence the review process. The factors investigated include the experience of the reviewer, team culture and communication, and consistent standards and guidelines. The majority of respondents suggested that these factors significantly affect code reviews, although some noted that the impact might be minimal. Overall, the data suggest that these factors do indeed influence code review outcomes.

4.2.2.1 How does the experience level or expertise of reviewers affect the outcome of code reviews?

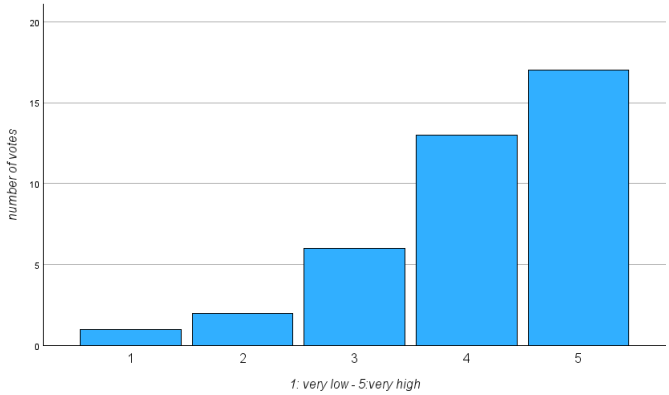

Figure 4.5: Expertise level

4.2.2.2 How does the Team Culture and Communication affect the collaboration in code reviews?

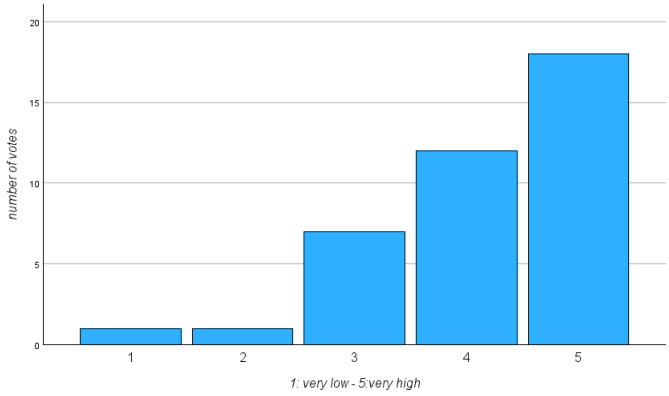

Figure 4.6: Team culture and communication

4.2.3 Collaboration and knowledge sharing

We posed two questions to determine whether code reviews encourage collaboration and knowledge sharing. By analyzing the data we have collected, we aim to answer these questions.

In Figures 4.8 and 4.9, we analyze whether code reviews enhance collaboration and knowledge sharing.
Figure 4.8 addresses whether pre-review discussions influence these aspects. The data reveal that 73.7% of respondents believe pre-review discussions significantly aid collaboration and knowledge sharing.
Figure 4.9 we asked if they agree that teaching the culture of taking criticism and learning from it, is important to enhance the collaboration and knowledge sharing.

4.2.2.3 Do you agree that Consistent Standards and Guidelines contribute to the effectiveness of code review processes?

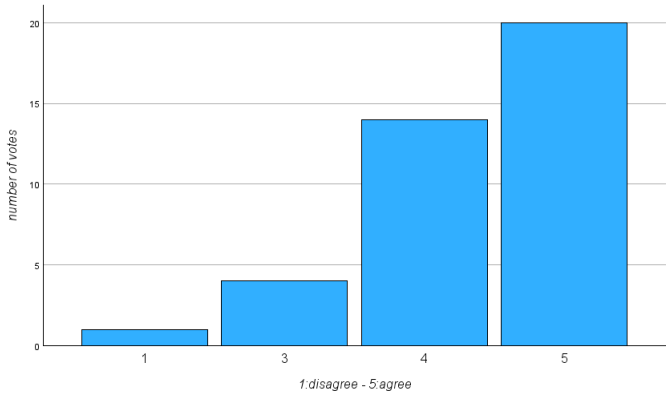

Figure 4.7: Standards and guidelines

The response was very clear that they agree and is important to enhance the collaboration and knowledge sharing

4.2.3.1 How does Pre-review discussions impacts the knowledge sharing between team members?

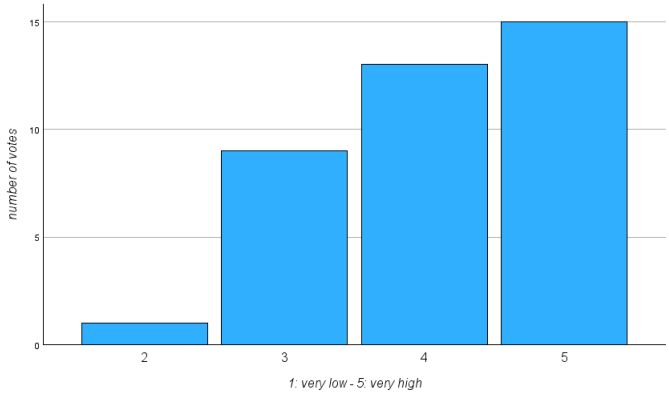

Figure 4.8: Pre-review discussions

4.2.4 Learning and skills development

We asked them if the feedback they received help them develop they coding skills. While the majority of respondents indicated that the feedback they received helped improve their development skills, as seen in Figure 4.10, it is important to note that a small number of participants gave lower ratings (2 or 3). This suggests that while most developers experience positive skill development through feedback, some may have had different experiences.

4.2.3.2 Do you agree that teaching the culture of taking critism and learning from it, is important to enhace the collaboration and knowledge sharing?

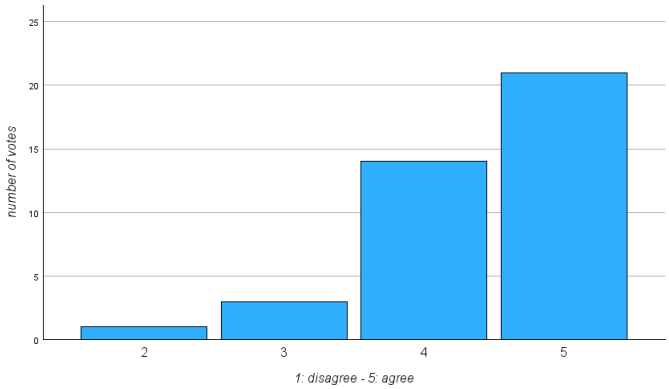

Figure 4.9: Critism culture

4.2.4.1 Do you agree that code review feedbacks that you get usually on your code helps you to develop your coding skills?

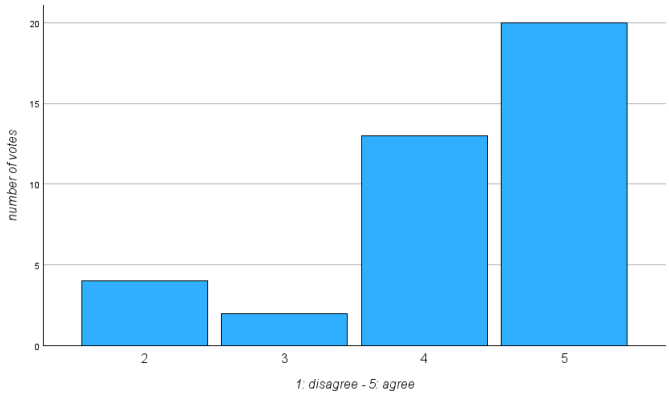

Figure 4.10: Coding skills

4.2.5 Would the industry be the same without code reviews

We wanted to see if code review was a big part of the industry or if the code review wasn't implemented would we see i difference. We asked this question in our survey, and it is clear to see that code reviews are a big part of the industry. In figure 4.11.

4.3 Answering the research questions

4.3.1 What factors contribute to the effectiveness of code review processes in software development?

By analyzing the responses from interviews and surveys, we can identify which factors most significantly contribute to the effectiveness of code reviews in software development. This analysis has established three main key factors: reviewer's expe-

4.2.5.1 Do you agree that industry of Software Development would not be the same without code reviews?

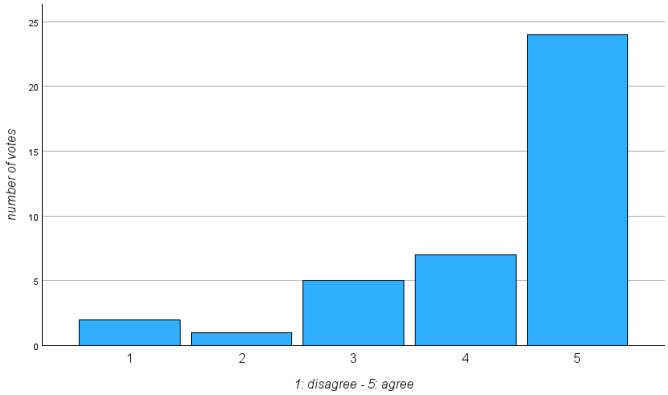

Figure 4.11: Software development industry viewpoint

rience, code complexity, and review practices, each encompassing numerous factors that influence overall effectiveness.

Reviewer's experience: The analyzed data identifies many advantages of having an experienced reviewer:

- Offering more insightful feedback
- Identify potential issues earlier
- Suggests practical improvements that enhance code readability and maintainability
- Accelerating the process of review

As well the experienced reviewer can have positive effects, the junior reviewer can also add some value by bringing fresh ideas and new perspectives.

Code complexity: Additionally, the code complexity being reviewed has an impact as well on the effectiveness of the process of code review, as the more complex the codebase, the more time and effort it takes to understand the code. However, some of the interviewers suggested adopting good practices such as simplifying pull requests which can help minimize the reviewed scope. Also, following the code standards and guidelines helps in keeping the code consistent and structured which significantly increases the readability and maintainability of the code.

Review practices: Review practices contribute as well to the effectiveness.
The interviewees have agreed on the following practices as they contribute the most to effectiveness:

- Pre-discussions of pull requests before start reviewing

- Dedicated time slots for code reviewing to maintain focus
- Code readability that helps in the faster understanding of the code
- Regular peer reviews
- Having checklists

4.3.2 How do code review practices impact collaboration and knowledge sharing among development team members?

To answer this question we are using the answers from the interviews and the data from the survey to point out which practices impact collaboration and knowledge sharing among the team and how it impacts.

Pre-Review Discussions: Both survey results and interviews highlight the importance of pre-review discussions. According to Figure 4.8, 73.7% of survey respondents believe these discussions significantly help collaboration and knowledge sharing. Interviewees underscored that discussing problems and solutions collectively before formal reviews ensures a unified understanding and strategy, fostering collaboration and reducing the likelihood of divergent feedback.

Feedback Quality and Constructive Criticism: The survey Figure 4.9 shows a strong agreement among participants that fostering a culture where criticism is viewed as a learning opportunity is crucial. Interviews complemented this by pointing out that high-quality feedback during reviews enhances learning, with experienced reviewers providing insights that help less experienced developers understand complex code aspects and improve their skills.

Experience Sharing and Mentorship: During code reviews, experienced developer provide guidance to less experienced team members. This mentorship is crucial for learning, as it helps newer developers improve their skills by learning directly from experts.

Implementing these practices ensures that code reviews are more than mere quality assurance measures, they become crucial learning experiences that significantly enhance team dynamics and knowledge dissemination. Promoting an inclusive environment where feedback is welcomed and valued, and facilitating open discussions about coding challenges and solutions, help teams significantly improve their collaboration, innovation, and productivity.

4.3.3 What is the role of feedback quality in code reviews, and how does it influence developer learning and skill development?

To address this question, we analyzed the responses from interviews with the survey data to show how the quality of feedback in code reviews impacts developer learning and skill development.

Impact of Feedback Quality: Both the survey and interviews highlight the critical role of feedback quality in improving the learning process for developers. According to the survey, majority of participants 84.6% combining ratings of 4 and 5 acknowledge that feedback from code reviews positively influences their coding skills development. This strong affirmation highlights the value of feedback in fostering professional growth.

Feedback Reception and Professional Growth: The way feedback is received also significantly impacts its effectiveness. Developers highlight the importance of approaching feedback with an open mind and a readiness to implement suggested improvements. Such an attitude not only improves individual coding skills but also contributes to the overall quality of the software developed.

Mentorship and Skill Enhancement: Experienced developers play a key role in this educational process, as highlighted in the interviews. By providing detailed, constructive feedback, experienced developers help junior developers improving their skills. This mentorship is crucial for sharing practical knowledge and best practices within the team.

The quality of feedback during code reviews is crucial for developers professional growth. Good feedback not only fixes immediate problems but also helps developers learn and improve their skills over time. By using a systematic and positive feedback approach, code reviews go beyond simple error checking to become key tools for experienced development and teamwork. By creating a workplace that values and effectively uses feedback, development teams can greatly improve their partners technical abilities and personal skills, which leads to more creativity and better results.

Chapter 5

Conclusions and Future Work

The findings of this study on this research provide important insights that both confirm and expand upon existing research.

Consistent with previous studies, our research underscores the critical role of code reviews in enhancing code quality and fostering knowledge sharing among developers. However, our study offers a precise perspective on knowledge sharing, reflecting the varied opinions found in related work. While some prior research did not observe significant knowledge transfer during code reviews, our results indicate that when done effectively, code reviews can indeed facilitate meaningful knowledge exchange. This, however, largely depends on the practices and culture within the development team.

Our research also highlights the significant factors that influence the success of code reviews, such as the experience of the reviewer, the complexity of the code, and the methods employed during the review process. Experienced reviewers are crucial for identifying both current and potential future issues in the code. The complexity of the code also plays a significant role, as more complex code requires greater time and effort to review, pointing to the need for improved methods to handle such challenges. For instance, practices like shorter pull requests, pre-review discussions, and adherence to coding standards, such as object-oriented programming principles, can enhance the effectiveness of the review process.

Where our study diverges from much of the related work is in its focus on the broader impact of code reviews on developer collaboration. Unlike earlier research that primarily addressed technical aspects, our findings highlight that code reviews are not just a tool for quality assurance but also a critical mechanism for improving teamwork. Practices like pre-review discussions and structured feedback sessions were found to significantly enhance collaboration and knowledge sharing, fostering a strong learning culture within teams.

Moreover, the quality of feedback during code reviews appeared as a essential factor. Effective feedback not only addresses immediate code issues but also supports the ongoing professional growth of developers. This continuous skill development enables developers to better meet the evolving demands of projects, further highlighting even more various benefits of code reviews.

In conclusion, while this study strengthens the established understanding of code reviews as a way to improve code quality and facilitate knowledge sharing, it also spots light on their broader effects on developer collaboration and skill development. These insights suggest that code reviews are a powerful tool for enhancing both individual developer growth and team dynamics, providing a richer understanding of their value in the software development process.

For future research :

Complexity of the Code: The impact of code complexity on the effectiveness of reviews received mixed opinions. Therefore, further research is necessary to conclusively determine how complexity affects the code review process.

Integration of AI in Code Reviews: With advancements in artificial intelligence, exploring how AI tools can be integrated into the code review process could provide significant benefits. AI could be used to automate aspects of reviews, predict areas of high risk, or even suggest refactoring opportunities. Investigating the effectiveness and limitations of AI in this context could improve the efficiency and quality of code reviews.

Sample Size and Generalizability: The small sample size in this study limits the generalizability of the findings. Future research should aim to include a larger and more diverse participant pool to ensure that the results are more representative of the broader software development commu

References

[1] A. Bacchelli and C. Bird, *Expectations, Outcomes, and Challenges of Modern Code Review.* IEEE, 2013.

[2] G. Bavota and B. Russo, *Four eyes are better than two: On the impact of code reviews on software quality.* IEEE, 2015.

[3] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, *The influence of non-technical factors on code review.* IEEE, 2013.

[4] M. Bhatia, *Your Guide to Qualitative and Quantitative Data Analysis Methods.* Humans of data, 2018.

[5] A. Bosu, M. Greiler, and C. Bird, *Characteristics of Useful Code Reviews: An Empirical Study at Microsoft.* Institute of Electrical and Electronics Engineers, 2015.

[6] J. Caulfield, *How to Do Thematic Analysis | Step-by-Step Guide & Examples.* Scribbr, 2019.

[7] M. Caulo, B. Lin, G. Bavota, G. Scanniello, and M. Lanza, *Knowledge Transfer in Modern Code Review.* IEEE, 2020.

[8] J. Czerwonka, M. Greiler, and J. Tilford, *Code Reviews Do Not Find Bugs. How the Current Code Review Best Practice Slows Us Down.* IEEE, 2015.

[9] L. Dong, H. Zhang, L. Yang, Z. Weng, X. Yang, X. Zhou, and Z. Pan, *Survey on Pains and Best Practices of Code Review.* IEEE, 2021.

[10] N. Fatima, S. Nazir, and S. Chuprat, *Individual, Social and Personnel Factors Influencing Modern Code Review Process.* IEEE, 2019.

[11] A. N. Ghazi, K. Petersen, S. S. V. R. Reddy, and H. Nekkanti, "Survey research in software engineering: Problems and mitigation strategies," *IEEE Access*, vol. 7, pp. 24 703–24 718, 2018.

[12] M. Höst and C. Johansson, *Evaluation of Code Review Methods through Interviews and Experimentation.* Journal of Systems and Software, 2000.

[13] O. Kononenko, O. Baysal, and M. W. Godfrey, *Code Review Quality: How Developers See It.* IEEE, 2016.

[14] MathWorks, *Code Review.* https://de.mathworks.com/discovery/code-review.html.

[15] L. Pascarella, D. Spadini, F. Palomba, M. Bruntink, and A. Bacchelli, *Information Needs in Contemporary Code Review.* Association for Computing Machinery, 2018.

[16] R. Streefkerk, *Primary vs. Secondary Sources | Difference & Examples.* Scribbr, 2018.

[17] D. O. Tuama, *The importance of Code Reviews.* Code Institute, 2024.

[18] K. Udoagwu, *6 Steps for a Solid Code Review Process.* Wrike, 2023.

[19] Łukasz Boruń, *What is a good code review process? Business benefits explained.* Miquido, 2023.

Faculty of Engineering, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden