Evaluation of Pair Programming

Praanto Samadder and Rushil Garg

May 16, 2024

Abstract

The goal of this research paper is to report on the effectiveness and efficiency of pair programming while participating in an agile team. The article uses personal accounts as material for analysis and presents the findings. The article also compares these findings with outcome from research on other agile teams.

Keywords: agile development, collaborative work, survey data analysis

1 Introduction

This report aims to assess the efficacy of pair programming within a group comprising individuals with varying levels of programming expertise. It delves into the outcomes and obstacles encountered during the development of a fact-checking tool utilizing natural language processing. Our evaluation of pair programming effectiveness is conducted through practice observation and comparison with findings from other researchers.

2 Background

Traditional development methods follow a sequential, linear approach with distinct phases like requirements gathering, design, implementation, testing, and maintenance. Examples include Waterfall and the V-Model, where each phase must be completed before moving on, leading to longer development cycles and less flexibility compared to Agile.[1]

Agile methods prioritize flexibility, collaboration, and customer feedback in iterative software development. They emphasize adaptive planning, continuous improvement, and delivering frequent, small increments of usable software through methodologies like Scrum, Kanban, and Extreme Programming (XP). Agile fosters responsiveness to change, customer involvement, and efficient delivery of value to stakeholders. [2]

The key differences lie in planning, flexibility, and iteration. Agile methods emphasize adaptability, collaboration, and delivering frequent increments of usable software, while traditional approaches favour a sequential, linear process

with less room for change once a phase is complete. Agile encourages continuous improvement and customer involvement, while traditional methods focus on following a predefined plan. Agile suits dynamic projects, while traditional methods may be better for stable, well-defined requirements.[3]

2.1 Pair programming

Pair programming is a collaborative software development technique where two programmers work together at one workstation. One programmer, known as the driver, writes the code, while the other, called the navigator, reviews each line of code as it is written. This dynamic duo continuously switches roles, enhancing code quality through real-time feedback, improved problem-solving, and knowledge sharing. Pair programming promotes teamwork, reduces errors, and fosters learning and creativity within the development process. [4] [5]

2.2 The project itself

This report encapsulates insights and learnings gleaned from an agile project focused on creating a fact-checking solution utilizing natural language processing model, specifically LLAMA-3-80B [6]. The objective of the project was to develop either a website enabling users to submit claims or a browser extension allowing users to scan articles for potential claims, with subsequent fact-checking functionality.

When scanning for articles, the backend would first use the LLM API to do claim extraction, then send a request to the retrieval API to retrieve relevant sources. Finally, the backend sends off those sources to the LLM API in order to run fact checking on them and returns the result to the front-end. The user sees these results as text highlights on the article that they ran the scan on.

3 Research Method

3.0.1 Identify Research Objectives

The primary objective of our research is to evaluate the extent to which pair programming has contributed to the success of our project. Pair programming, a collaborative software development technique where two programmers work together at one workstation, has been implemented within our team as a means to improve code quality, increase productivity, and enhance overall project outcomes. Our research aims to provide evidence-based insights into the efficacy of pair programming in achieving these objectives.

3.0.2 Research Questions

1. To what extent has pair programming influenced the efficiency and productivity of our development process?

2. How has pair programming affected the quality and correctness of the code produced?

3. What are the perceived benefits and challenges of pair programming as reported by team members?

3.0.3 Research Methodology: Surveys

To address these research questions, we have chosen to utilize surveys as our research methodology. Surveys offer a structured approach to collecting feedback and insights from team members engaged in pair programming. By administering surveys, we can gather quantitative and qualitative data to assess the impact of pair programming on various aspects of our project.

3.0.4 Data Analysis and Interpretation

Once survey responses have been collected, we will conduct a comprehensive analysis to identify trends, patterns, and correlations in the data. Quantitative data will be analyzed using statistical techniques to measure the impact of pair programming on project outcomes. Qualitative responses will be coded and thematically analyzed to extract key themes and insights.

3.1 Expected Outcomes

Through this research, we anticipate gaining a deeper understanding of how pair programming has influenced our project. We expect to uncover insights into the benefits and challenges associated with pair programming, as well as its overall impact on productivity, code quality, and team dynamics. These findings will inform decision-making and provide valuable guidance for optimizing our development process in future projects.

3.2 Evaluation Criteria

In evaluating our practice of pair programming, we utilize the following criteria:

3.2.1 Efficiency

This criterion assesses how effectively pair programming enhances productivity and reduces development time. Efficiency encompasses factors such as code quality, task completion rates, and overall team productivity. By monitoring efficiency, we aim to identify areas for optimization and improvement in our development process.

3.2.2 Correctness of Code

Ensuring the correctness of the code is paramount in software development. This criterion evaluates the accuracy and reliability of the code produced through

pair programming. It considers factors such as adherence to coding standards, absence of bugs and errors, and the effectiveness of testing procedures. By prioritizing correctness, we aim to deliver high-quality software that meets the needs and expectations of our stakeholders.

4 Project Results

We received the following responses to our Survey Questions:

4.1 To what extent has pair programming influenced the efficiency and productivity of our development process?

- Praanto: Pair programming has significantly improved our performance and has been instrumental in leveling the skill gap between several members of the team. While it may sometimes seem cumbersome, the collaborative nature of pair programming fosters knowledge sharing and faster problem-solving, ultimately enhancing our overall efficiency.

- Kristians: Despite the perception that pair programming halves the number of tasks that can be done simultaneously, I've observed that the time to complete a task is usually lessened. This is because more otherwise-missable issues are caught during the development process, rather than during testing. The collaborative nature of pair programming enables us to catch and address potential issues in real-time, leading to more efficient development cycles.

- Rushil: Although pair programming may initially slow down more experienced developers, it allows for more tasks to be parallelized as we progress. This iterative improvement in task parallelization ultimately enhances our overall efficiency. Additionally, pair programming facilitates continuous learning and skill development among team members, further contributing to our productivity in the long run.

- Zaid: Personally, I prefer not working simultaneously in real-time during pair programming sessions. Instead, I believe code reviews are more effective, as they allow for a more thorough examination of the code without the potential distractions or conflicts that may arise during real-time collaboration.

4.2 How has pair programming affected the quality and correctness of the code produced?

- Praanto: Pair programming has indeed reduced our efficiency, but it has significantly increased the accuracy and reliability of our code. By having two sets of eyes on the code as it's being written, we catch and address

potential issues earlier in the development process, resulting in code with fewer critical bugs.

- Kristians: Overall, the practice of pair programming has led to code with fewer critical bugs. Having someone else look over your code in real-time helps catch issues that might have otherwise been missed, ultimately resulting in higher-quality code. This collaborative approach to coding ensures that potential errors are identified and resolved promptly, contributing to the overall correctness of our codebase.
- Rushil: Pair programming plays a crucial role in improving the quality of our code. By working collaboratively, any mistakes or inconsistencies are more likely to be caught at an earlier stage, reducing the likelihood of introducing bugs into the codebase. This proactive approach to code review and validation enhances the overall correctness and reliability of our code.
- Zaid: As I did not participate in pair programming sessions, I do not have a particular opinion on how it has affected the quality and correctness of the code produced. However, I acknowledge that the collaborative nature of pair programming can offer valuable insights and opportunities for code improvement.

4.3 What are the perceived benefits and challenges of pair programming?

- Praanto: While pair programming facilitates quicker error resolution, it can sometimes be cumbersome to coordinate and collaborate effectively. However, the benefits of pair programming, such as knowledge sharing and skill leveling, outweigh these challenges in the long run.
- Kristians: The benefits of pair programming include increased accuracy and efficiency, particularly when the pair consists of compatible personalities. However, challenges may arise if the pair consists of incompatible personalities, which can hinder efficiency. Despite these challenges, the overall benefits of pair programming in terms of code quality and collaboration make it a valuable practice for our team.
- Rushil: Although pair programming may lead to a decrease in efficiency, especially if the programmers have significant differences in skill sets, it offers numerous benefits such as improved code quality and error detection. The collaborative nature of pair programming fosters teamwork and knowledge sharing, ultimately contributing to the success of our projects.
- Zaid: As I did not engage in pair programming, I do not have a particular opinion on the perceived benefits and challenges of this practice. However, I recognize that pair programming can offer advantages in terms of code quality and collaboration, while also presenting challenges related to coordination and compatibility among team members.

5 Analysis

5.1 Efficiency and Productivity

5.1.1 Findings

The responses reflect a nuanced perspective on the impact of pair programming on efficiency and productivity within our team. Praanto and Rushil emphasize the tangible improvements in performance and efficiency brought about by pair programming, highlighting its role in leveling the skill gap among team members. Their experiences underscore the collaborative nature of pair programming, which facilitates knowledge sharing and faster problem-solving. However, Kristians' observation introduces a contrasting viewpoint, noting that while pair programming may halve the number of tasks that can be done simultaneously, it often leads to a reduction in the time taken to complete tasks due to the early detection of issues. This perspective emphasizes the importance of quality over quantity and highlights the preventive nature of pair programming in catching potential errors before they escalate. Conversely, Zaid's preference for code reviews over real-time collaboration reveals a potential weak point in the practice, suggesting a lack of alignment or understanding regarding the benefits of pair programming among team members.

5.1.2 Weak Points and Lessons Learned

The mixed perceptions regarding efficiency and productivity suggest a need for clearer communication and alignment within the team regarding the objectives and benefits of pair programming. While some team members recognize its value in improving performance and efficiency, others may need further education or clarification on its role in enhancing code quality and collaboration. Addressing these gaps through targeted training, team discussions, and sharing success stories can help build consensus and enthusiasm for pair programming among all team members. [7] Additionally, incorporating feedback mechanisms to regularly assess and adjust the pair programming process can help identify and address any inefficiencies or challenges that arise.

5.2 Quality and Correctness of Code

5.2.1 Findings

The responses generally indicate a positive impact of pair programming on the quality and correctness of the code produced. Praanto acknowledges a reduction in efficiency but emphasizes the increased accuracy and reliability of the code. Kristians and Rushil highlight the benefits of having another set of eyes on the code, leading to fewer critical bugs and improved code quality. However, Zaid's lack of participation in pair programming sessions limits the scope of this assessment.

5.2.2 Weak Points and Lessons Learned

While the majority of team members recognize the benefits of pair programming in improving code quality, there may be challenges in ensuring consistent participation and engagement from all team members. Encouraging active involvement in pair programming sessions and providing opportunities for feedback and reflection can help address any concerns or reservations.

5.3 Perceived Benefits and Challenges

5.3.1 Findings

The perceived benefits and challenges of pair programming identified by team members provide valuable insights into the practice's overall effectiveness within our team. Praanto and Rushil highlight the benefits of quicker error resolution and increased accuracy resulting from pair programming, underscoring its role in improving code quality and reliability. However, they also acknowledge potential challenges such as the perceived cumbersome nature of pair programming, indicating room for improvement in the process's implementation and execution. Kristians further emphasizes the importance of compatibility and effective communication in pair programming partnerships, suggesting that the success of the practice relies heavily on interpersonal dynamics and team cohesion. Conversely, Zaid's lack of participation in pair programming limits the diversity of perspectives on this criterion, posing a potential weak point in the assessment's comprehensiveness.

5.3.2 Weak Points and Lessons Learned

The challenges identified, such as coordination issues and compatibility concerns, highlight the importance of fostering a supportive and collaborative team environment to maximize the effectiveness of pair programming. Addressing these challenges may require strategies such as team building activities, communication workshops, and regular feedback sessions to strengthen interpersonal relationships and promote effective collaboration. [8] Additionally, providing opportunities for all team members to participate in pair programming activities and share their experiences can help ensure a more comprehensive and inclusive assessment of its benefits and challenges. By addressing these weak points and leveraging lessons learned, we can further optimize the practice of pair programming and enhance its overall impact on our team's success.

6 Conclusion

In conclusion, our investigation into the practice of pair programming has provided valuable insights into its impact on our development process. Through the perspectives shared by team members, we've identified both strengths and

areas for improvement in our implementation of pair programming. Pair programming has emerged as a promising approach to enhancing code quality, improving efficiency, and fostering collaboration within our team. The firsthand experiences of Praanto, Kristians, and Rushil attest to the tangible benefits of pair programming, including quicker error resolution, increased accuracy, and knowledge sharing. These findings align with existing research and underscore the potential of pair programming as a valuable tool in software development. However, our analysis also highlights challenges such as coordination issues, compatibility concerns, and differing perceptions among team members. Addressing these challenges will be crucial to optimizing the effectiveness of pair programming and maximizing its impact on our project outcomes. Looking ahead, future work will focus on refining our pair programming practices, addressing identified weak points, and further integrating pair programming into our development process. This may involve implementing targeted training programs, facilitating open communication and feedback channels, and fostering a culture of collaboration and continuous improvement within our team. In conclusion, pair programming holds tremendous potential as a strategy for enhancing our development process and achieving our project objectives. By embracing its principles, addressing challenges, and continuously refining our approach, we are well-positioned to unlock the full benefits of pair programming and drive success in our future projects.

References

[1] Shamsulhuda Khan, A Comparative Study of Agile and Waterfall Software Development Methodologies, *DOI: 10.48175/IJARSCT-5696*

[2] Cockburn, Alistair, and Laurie Williams. "The costs and benefits of pair programming." *Extreme programming examined 8 (2000): 223-247.*

[3] Jordan B Barlow, et. al, Overview and Guidance on Agile Development in Large Organizations, *Volume 29, Article 2, pp. 25-44, July 2011*

[4] Nawrocki, Jerzy, and Adam Wojciechowski. "Experimental evaluation of pair programming." *European Software Control and Metrics (Escom) (2001): 99-101.*

[5] Joe E Hannay, The effectiveness of pair programming: A meta-analysis, *DOI: 10.1016/j.infsof.2009.02.001*

[6] Meta, Introducing Meta Llama 3: The most capable openly available LLM to date, last accessed May 16 2024 at 15:40

[7] Rinaily Bonifacio, The Skills Gap Challenge: Understanding and Overcoming It, last accessed May 16, 2024 at 22:44

[8] Sadman Sakib Khan, Benefits Of Having An Agile Team Building Your Digital Product