Evaluation of Pair Programming Practice

IV1303 Modern Mjukvaruutveckling

Hiba Qutbuddin Habib

hibaqh@kth.se

School of Electrical Engineering and Computer Science (EECS)
Kungliga Tekniska Högskolan
Sverige

May 2023

Abstract

In today's society, technology and technological development have become important cornerstones of societal functioning. A large part of our lives today is dependent on technology, and software development has become very common. Numerous methods have been developed to improve and streamline this development, and Pair Programming is an example.

This research report is about pair programming. The purpose is to study both the advantages and disadvantages of this method. This study is based on work in a software project where pair programming was applied extensively. The study aims to answer the following question: Is the Pair Programming method effective and good enough to improve workflow and accuracy in a project?

The study is based on the following criteria: Driver/Navigator, Effectiveness and Collaboration. The result showed that the use of the method contributes to increased efficiency, and the use of pair programming leads to enhanced knowledge sharing among the team compared to individual programming.

Keywords

Project, Software development, Sprints, Driver, Navigator.

Table of Content

1 Introduction **3**

2 Background **4**

2.1 Traditional Development methods . . . 4
2.2 Agile development methods . . . 4
2.3 Differences between traditional and agile development methods . 5
2.4 Software development practices . . . 5
2.5 Pair Programming . . . 6
2.6 Project . . . 6

3 Research Method **7**

3.1 Research Phases . . . 7
3.1.1 Literature review . . . 7
3.1.2 Daily & Weekly summaries . . . 7
3.1.3 Final compilation & Comparison with other research . . . 8
3.2 Evaluation Criteria . . . 8
3.2.1 Driver/Navigator . . . 8
3.2.2 Effectiveness . . . 8
3.2.3 Collaboration . . . 8

4 Project Results **9**

4.1 Driver/Navigator . . . 9
4.1.1 Driver . . . 9
4.1.2 Navigator . . . 9
4.2 Effectiveness . . . 10
4.3 Collaboration . . . 12

5 Result analyze **14**

5.1 Driver/Navigator . . . 14
5.1.1 Driver . . . 14
5.1.2 Navigator . . . 15
5.2 Effectiveness . . . 15
5.3 Collaboration . . . 16

6 Conclusion and Future Work **18**

7 References **19**

1 Introduction

The field of technical development has become incredibly expansive, with a strong emphasis on software development. As this industry continues to gain prominence worldwide, numerous methods have been developed to complement it, including Scrum, Test-first development, Refactoring, Pair programming, and more. These methods are designed to simplify and optimize work processes while fostering creativity and striving for the best possible solutions and innovations.

In this report, we delve into the methodology known as Pair Programming. The effectiveness of this approach was put to the test in a month-long software project. Our study draws upon the project's outcomes, supplemented by an analysis of research articles and case studies from various companies that have successfully employed Pair Programming in their workflows. The primary objective of this study is to comprehensively evaluate the benefits and drawbacks associated with this method.

What is Pair programming? Pair programming is a method that has gained increasing popularity in recent times in the context of software-based projects. The Pair programming method involves two programmers collaborating and working together on the same computer to develop or write code. Opinions about this method have been somewhat divided. Some are against it, as they believe it to be a waste of resources to have two individuals working on the same task, using the same computer and code. However, there are those who support the method and argue that Pair programming leads to a significant boost in productivity. They believe that the collaborative nature of Pair programming enhances problem-solving, code quality, and knowledge sharing, resulting in faster and more efficient development[1].

In this report, we delve into a comprehensive discussion of both the advantages and disadvantages of the method. A significant emphasis will be placed on exploring the aspect of efficiency. Specifically, we aim to determine whether the implementation of Pair Programming leads to enhanced effectiveness in the work process.

The report is structured as follows: following this introductory section, section two provides a background on the subject matter. Section three encompasses the research methodology employed, while section four presents the obtained results. These findings are subsequently analyzed in section five, and finally, a concise summary along with the reference list is presented in section six and seven.

[1]`https://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF`. A.Cockurn, L.Williams

2 Background

This section primarily focuses on the theoretical aspect of the subject. Subsequently, two distinct software development methods are presented: the traditional development method, which follows a 'waterfall model,' and the Agile development method, which incorporates 'Scrum.' A thorough comparison is made between these two approaches. Furthermore, the method discussed in the report is described, along with the project that was examined in relation to it.

2.1 Traditional Development methods

The traditional development method relies on a sequential series of steps. The key concept behind this approach is the ability to predict project outcomes from the project's inception. To accommodate this, comprehensive planning of the entire project cycle is carried out at the early stages. Within the traditional method, there are different development models, one notable example being the waterfall method.

The Waterfall model is a sequential process consisting of four distinct phases[2].The first phase focuses on building the requirements specification, followed by the design phase in the second stage. The development process commences in the third phase, and finally, the testing phase ensues.

The objective of the testing phase is to ensure that the product meets the specified requirements. Once this criterion is fulfilled, the product is deemed ready for customer use [3].

2.2 Agile development methods

The Agile development method differs significantly from the traditional approach. Agile methodology places a strong emphasis on continuous development, wherein project members follow a cycle encompassing planning, execution, and evaluation. These iterative steps are repeated across different phases of the project, fostering adaptability and flexibility.[4].

Scrum is a renowned model within the Agile development methodology. Its core concept involves breaking a project into distinct sprints, each adhering to a consistent structure. At the start of each sprint, planning takes place to establish the goals to be accomplished by its end. Upon completion of a sprint, a retrospective is conducted—a meeting where all team members reflect on the finished sprint and collectively determine areas for improvement in the upcoming sprint. Additionally, each sprint concludes with a demo, where the achievements and outcomes of the sprint are showcased. [5]

[2] `https://wiki.edunitas.com/IT/en/114-10/Waterfall_22494_eduNitas.html`

[3] `http://ku-fpg.github.io/files/agile-traditional.pdf`. Y.Leau, W.K.Loo, W.Y.Tham, S.F.Tan

[4] `https://www.sciencedirect.com/science/article/pii/S0164121212000532`. T.Dingsoyr, S.Nerur, V.Balijepally, N.B.M

[5] `https://ieeexplore.ieee.org/abstract/document/8229928`. A.Srivastava,

2.3 Differences between traditional and agile development methods

The Agile and traditional methods differ significantly. The traditional method relies on a predictive approach, while Agile is based on adaptive software development.

These methods vary in several aspects, including their working approaches. The traditional approach adheres to a detailed plan with an exhaustive list of characteristics and tasks to be executed throughout the project's lifecycle. In contrast, Agile employs an adaptive approach, foregoing a comprehensive plan and instead utilizing continuous partial plans throughout the project. Agile teams are more adept at adapting to dynamic changes in product requirements due to ongoing product testing during the project cycle. Consequently, Agile facilitates easier and more cost-effective risk management compared to the traditional method.

Communication within Agile projects is typically stronger, as team members often work in close proximity and maintain continuous contact with the customer throughout the project. The Agile method boasts greater flexibility and superior risk management, leading to reduced costs. However, there are situations where the Agile method may not be applicable. For instance, Agile functions optimally with small teams (around 10 members or fewer) and smaller projects that require fewer resources. In contrast, the traditional method is better suited for larger-scale projects that demand improved organization.

To successfully implement Agile practices within an organization, a willingness to embrace change and acceptance of new methodologies is crucial. Without these elements, adopting the Agile method becomes challenging[6].

2.4 Software development practices

Software development practices encompass the methods and processes employed in contemporary software development. Their aim is to enhance efficiency, productivity, and overall performance within the industry. These practices strive to yield positive outcomes, benefiting both the work environment and the professional growth of individuals. However, the extent of their effectiveness varies among different workplaces, influenced by the nature of their work and the employees' receptiveness to these methodologies[7].

A list of some Software development practices is given below, the list is not in any particular order:

- Code refctoring
- Code review
- Pair Programming

S.Bhardwaj,S.Saraswat

[6]`http://www.revistaie.ase.ro/content/68/0620-20Stoica,20Mircea,20Ghilic.pdf`. M.Stoica, M.Mirecea, B.Ghilic-Micu

[7]`https://en.wikipedia.org/wiki/Software_development_process`.

- Retrospektive
- Unit testing
- Naming conventions

The list above includes several examples of software development practices. The report specifically focuses on one of those practices mentioned earlier, which is ”Pair Programming.”

2.5 Pair Programming

Pair programming is an agile software development technique where two developers collaborate on the same computer to design, code, and test. It involves two distinct roles: the ”driver” and the ”navigator.” The driver is responsible for actively writing the code, while the navigator focuses on the overall direction of the program, reviewing the code and ensuring its quality.

Role switching is a fundamental aspect of pair programming, as it promotes knowledge sharing and prevents fatigue. By periodically switching roles, both developers gain a comprehensive understanding of the codebase and actively contribute their expertise.

The primary goal of pair programming is to enhance collaboration and problem-solving. By working together, developers can tackle complex challenges more effectively than they would individually. Effective communication is key to successful pair programming, enabling seamless exchange of ideas, clarifications, and continuous feedback[8].

2.6 Project

The studies in this report are partially based on a four-week project undertaken by the author. The project follows the Scrum software development practice and incorporates several other practices, with Pair Programming being the most extensively used. This choice was driven by the diverse skill levels of team members, who collectively agreed on Pair Programming’s central role in the project.

The project’s objective was to create a web application designed to assist novice students at KTH with programming. The idea originated from the team members’ firsthand experience, as they found the Programming 1 course particularly challenging for students without prior programming knowledge. Thus, the aim was to develop a web application that adopts a student-centric approach to learning, intending to simplify programming and enhance its enjoyment for others.

[8]`https://www.techtarget.com/searchsoftwarequality/definition/Pair-programming`. Alexander S,Gilllis

3 Research Method

In this section, the method used in the investigation of the subject "Pair Programming" is presented.

3.1 Research Phases

During the project, an investigation of Pair Programming was conducted. The process consisted of five steps: literature review, daily summaries, weekly summaries, final compilation, and comparison with other research in the field. The figure below illustrates the different steps in the process:

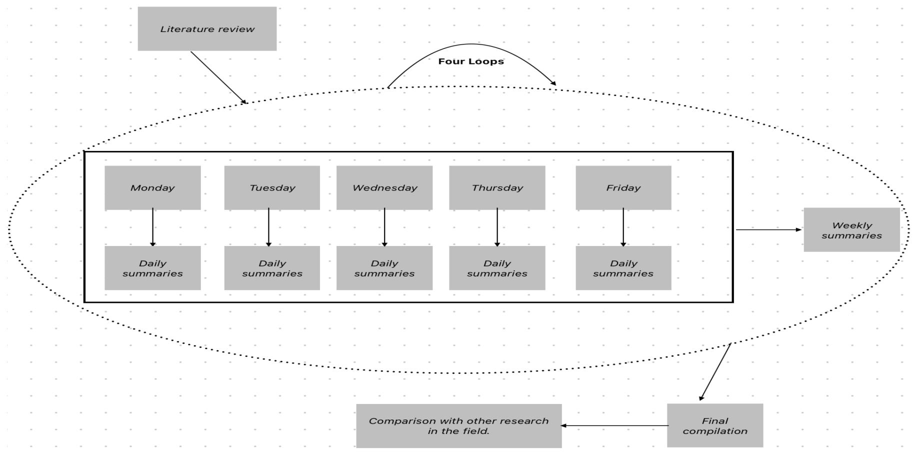

Figure 1: Illustration of the different steps in the process

3.1.1 Literature review

Before the project started, a literature study was conducted on the topic of 'Pair Programming.' The focus was to gain insight and a general understanding of the subject, which was helpful in writing the background section of the report. It also facilitated the definition of Evaluation Criteria and comparison of the investigation during the project.

3.1.2 Daily & Weekly summaries

The project consisted of 4 sprints, each lasting 5 days. At the end of each day, a brief summary called "Daily summaries" was conducted, where the results of the work using the pair programming method were compiled, and a short reflection was noted down. At the end of each sprint, a joint "Weekly summaries" was also conducted with other team members, which was also referred to as "Retrospective".

3.1.3 Final compilation & Comparison with other research

At the end of the project, a final compilation was conducted, combining all the Daily & Weekly summaries. The final results were then compared with other research in the field.

3.2 Evaluation Criteria

In order to conduct a concrete and effective investigation of the Pair Programming method, a set of additional criteria was defined. These criteria were measured and followed throughout the project. The results of this investigation are also based on the following criteria:

3.2.1 Driver/Navigator

This criterion is based on the driver/navigator's experience of pair programming. It evaluates the role and perception of the method from their perspective and experience, comparing it to other studies.

3.2.2 Effectiveness

This criterion relates to the effectiveness of pair programming. Is it time-saving? In the project, both single programming and pair programming were performed, and in order to measure this criterion, the time for each working method is compared. Throughout the project, there were 4 different sprint planning sessions where an estimation for the time each task may require was established. These values will also be compared with the actual time the tasks took when using pair programming.

3.2.3 Collaboration

The focus of this criterion was on the effectiveness of collaboration among team members when using the Pair Programming method. The team members had varying levels of knowledge regarding the programming languages and building different components required for a functioning backend. This resulted in individuals with different levels and with same levels of experience and knowledge working together through Pair Programming. Consequently, members who possessed more knowledge about a specific aspect needed to provide a brief overview to those with less knowledge before starting Pair Programming. To evaluate this criterion, we measure the knowledge level of the pairs engaging in pair programming both at the start and at the end of the project, and analyze the synchronization time between pairs with varying levels of expertise, comparing them to pairs with same levels of knowledge.

4 Project Results

In this section of the report, the results of the investigation conducted during a four-week-long project are presented. The results are presented under headings 4.1-4.3, following the same order as the evaluation criteria.

4.1 Driver/Navigator

4.1.1 Driver

	Week 1	Week 2	Week 3	Week 4
Monday	-	7	-	6
Tuesday	-	9	7	7
Wednesday	8	8	7	-
Thursday	8	8	8	-

Table 1: The table shows driver ratings from 1 to 10

The result for this criterion shows how the driver experienced pair programming over a four-week period. Pair programming was applied eleven times during the project, and the driver had to rate the method on a scale of 1 to 10, where 1 means very poor and 10 means very good.

The table above summarizes the driver's ratings for all pair programming sessions. From the table, we can see that the driver has given a minimum score of sex. The compilation of results in the table yields an average value of approximately 7.55 out of 10.

4.1.2 Navigator

	Week 1	Week 2	Week 3	Week 4
Monday	-	5	-	8
Tuesday	-	6	7	6
Wednesday	4	7	8	-
Thursday	6	7	8	-

Table 2: The table shows navigator ratings from 1 to 10

To compile a concrete result for the experience of implementing the pair programming method, in addition, the navigator was also asked to rate the experience and knowledge of pair programming from a navigator's perspective. The navigator's rating was examined in the same way as the aforementioned criteria for the driver.

Table 2 above shows the results of the rating. The lowest rating indicated by the navigator was four, and a compilation of all the ratings presented in the table yields an average value of 6.55 out of 10.

4.2 Effectiveness

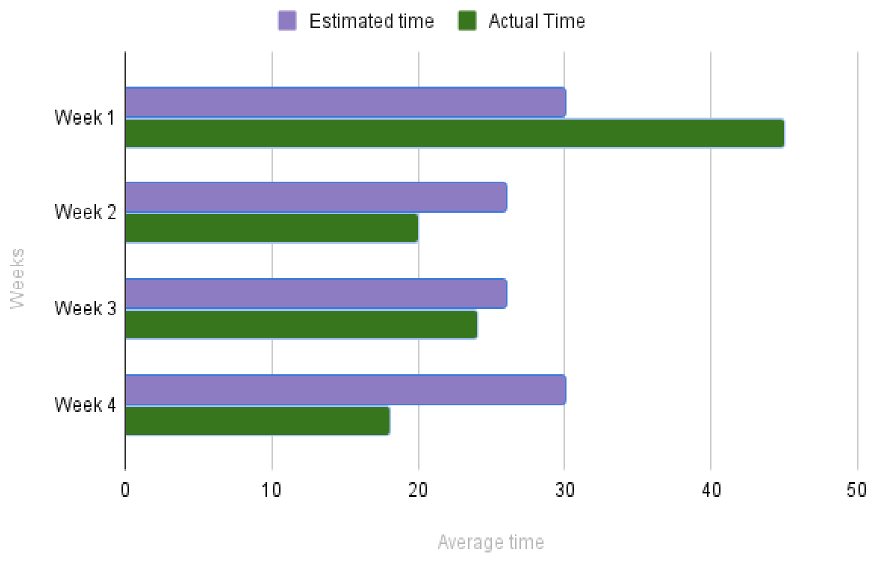

Figure 2: The diagram shows a comparison between estimated time and actual time to complete a task

The project's efficiency was assessed as one of the criteria. Time estimation was carried out for the work conducted throughout the project. Sprint planning sessions, held at the start of each sprint, involved estimating time alongside other team members. At the end of each sprint, a retrospective meeting was held to summarize the week, discuss areas for improvement in the upcoming sprint, and evaluate the accuracy of time estimation and the team's progress in line with the plan.

The diagram above illustrates a comparison between the estimated and actual time taken for each week's work. The data shows that the initial week required more time than estimated, while subsequent weeks demonstrated improved efficiency, with work being completed in less time than anticipated. This indicates the effectiveness of pair programming in achieving this efficiency.

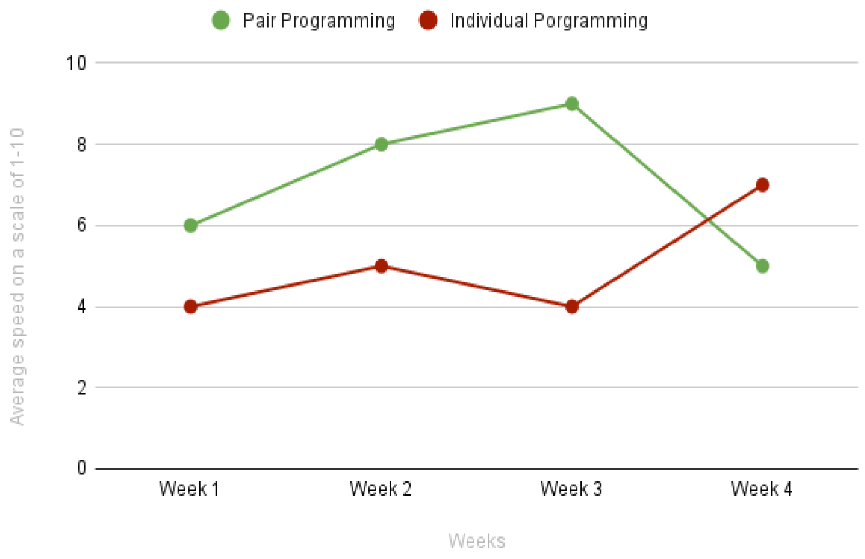

Figure 3: The diagram shows average speed on a scale of 1-10 for performing a task using both Pair and Individual programming

To obtain tangible efficiency results, the speed of task execution is measured for both individual programming and pair programming. Speed is rated on a scale from 1 to 10, with 1 indicating a very slow pace and 10 denoting a highly rapid pace.

The diagram above illustrates the survey findings. It reveals that pair programming consistently delivered a high execution speed throughout the majority of the project. However, towards the project's conclusion, individual programming exhibited a higher speed.

4.3 Collaboration

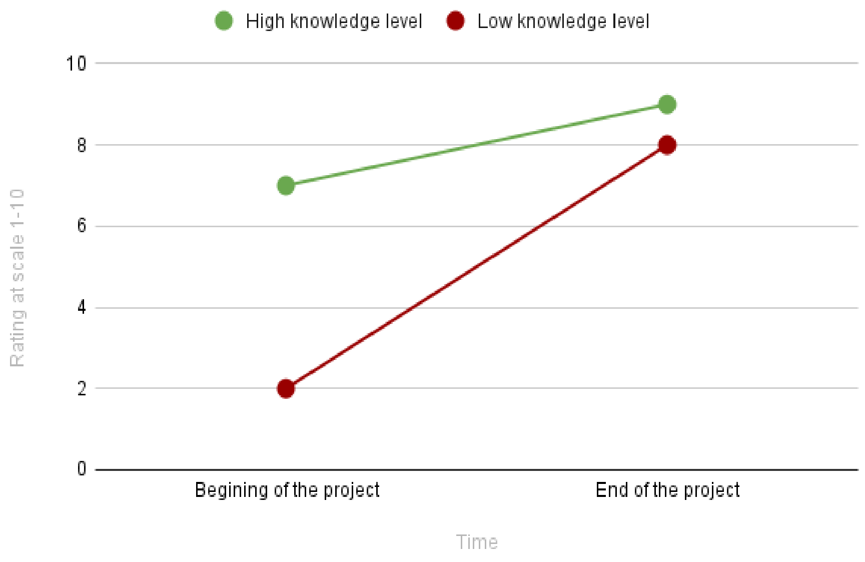

Figure 4: The diagram illustrates the difference in knowledge level at the beginning and end of the project on a scale from 1 to 10

The final criterion for studying pair programming is collaboration. This criterion focuses on investigating the development of team members based on their knowledge level and the time it takes for synchronization within the pairs.

To assess the development of team members, their estimated knowledge levels were recorded at the beginning and end of the project. The team was divided into two groups: one with a low knowledge level in the programming language and method used for creating a functional backend, and another with a high knowledge level. Knowledge levels were rated on a scale of 1-10.

The diagram above illustrates the compiled results for both groups' knowledge levels before and after the project. It shows an increase in knowledge level for both groups.

Pair programming was conducted in two different ways: some groups consisted of pairs with similar knowledge levels, while others comprised pairs with varying knowledge levels. Synchronization within the groups was measured at the project's outset and during the shifting of pairs throughout the projects.

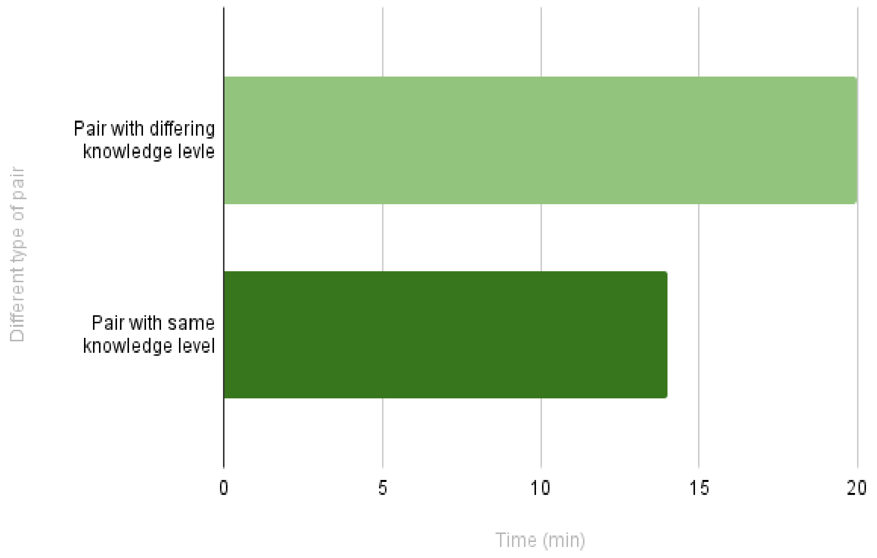

Figure 5: The diagram displays the average time it took for different pairs to synchronize

The diagram above shows the average time taken by the different pairs to synchronize. The findings reveal that pairs with similar knowledge levels required less time compared to pairs with disparate knowledge levels.

5 Result analyze

In this section, the results of the report are analyzed in comparison to previous research in the field.

5.1 Driver/Navigator

The objective of this criterion was to investigate the overall experience of both the driver and navigator in pair programming. Taking into account their perspectives and impressions, they assessed the effectiveness of the pair programming method in various project scenarios. The findings are detailed in sections 4.1.1 and 4.1.2.

5.1.1 Driver

The drivers' ratings for this method averaged at 7.55 out of 10, with the lowest rating given being 6. Considering the overall average and the lowest score reported by the drivers regarding their experience with the method, it appears that the results are generally positive. This suggests that drivers have found value in utilizing this approach.

To gain a deeper understanding of these ratings, drivers were interviewed to gather their feedback. During the interviews, drivers expressed various perspectives highlighting both the advantages and disadvantages of the method, which influenced their ratings.

According to drivers, the method offers several benefits. Firstly, it enhances work efficiency by facilitating quicker problem-solving through the introduction of a second person who can provide alternative viewpoints. This prevents getting trapped in a singular perspective. Additionally, drivers mentioned that having another pair of eyes reduces pressure and helps in identifying and rectifying coding errors.

However, drivers also pointed out a few drawbacks. They mentioned that it can be challenging when the navigator lacks the same level of expertise, leading to an uneven distribution of workload. Drivers feel the need to be vigilant about all aspects of the work while simultaneously explaining things to the navigator.

In an article authored by Alistair Cockburn and Laurie Williams, they highlight the value of pair programming. According to their findings, participants have reported a heightened sense of enjoyment in their work when engaging in pair programming[9]. The driver, responsible for actively writing the code, benefits from this collaborative approach by having errors in the code promptly detected during the coding process, rather than after its completion. Additionally, problem-solving becomes more efficient as participants work together in pairs, which has been positively received by the driver as well.

[9] `https://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF`. A.Cockburn, L.Williams

5.1.2 Navigator

The Navigator method received an average rating of 6.55 out of 10, indicating a moderately positive response. The lowest rating reported was 4. Analyzing the survey results presented in Table 4.1.2, it becomes apparent that the navigator received the lowest ratings during the initial sprint. However, as the subsequent sprints progressed, the navigator's ratings improved significantly. Overall, the findings suggest that the navigator had a largely positive experience with pair programming.

To gain deeper insights into the navigator's experience and ratings, an interview was conducted. The navigator's feedback revealed that participants with limited knowledge found pair programming to be an incredibly enriching opportunity. However, a notable drawback mentioned by the interviewees was the novelty of assuming the role of a navigator/observer, which posed challenges for most of them. Consequently, some participants felt they were not actively contributing and their involvement in the work was diminished.

In an article authored by Danielle L. Jones and Scott D. Fleming, the potential of collaborative learning in pair programming is emphasized. Partners in this approach often share their programming knowledge and exchange insights about the tools employed. This notion is reaffirmed in the present report, as the navigator experienced pair programming as an immensely enriching opportunity. However, the article also addresses a challenge encountered when the navigator falls behind, struggling to keep pace with the driver's code. This divergence arises from variances in knowledge between the navigator and the driver. Participants who assumed the navigator role and were interviewed expressed that, due to the perceived subordinate position, they sometimes feel they contribute less to the overall work[10].

5.2 Effectiveness

The objective of this criterion was to compare the effectiveness of pair programming with individual programming. We measured this criterion in two ways. First, we examined the estimated and actual times for completing tasks during each sprint planning. The results are depicted in Figure 2, Section 4.2. In the initial sprint, the actual time exceeded the estimated time, possibly due to changes in the project approach made by the group two days after starting the sprint. This led to a substantial amount of rework. However, in the subsequent sprints, the graph demonstrates that the actual time for completing tasks was shorter than the estimated time, indicating increased efficiency.

The second measure of effectiveness was the speed rating, ranging from 1 to 10, for task execution in both pair programming and individual program-

[10]`https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=6645252&tag=1`. Danielle L.Jones, Scott D.Fleming

ming. Figure 3, Section 4.2, presents the results. The graph suggests that pair programming generally resulted in higher speed throughout most of the project. Towards the end, however, individual programming became more efficient. This can be attributed to the team's focus on content development for the website during the last week/sprint, which did not require pair programming and thus benefited from individual programming.

The results of this criterion generally demonstrate that working with pair programming yields more efficient outcomes. In the report titled "Increasing Quality with Pair Programming" by Kim Nilson, the method of pair programming is thoroughly examined. The findings reveal that implementing this method can lead to a time-saving of up to half the usual time. Pair programming proves particularly advantageous in swiftly solving complex problems, facilitating more efficient code implementation, and enhancing overall quality. However, it should be noted that the effectiveness of pair programming diminishes when dealing with simpler tasks[11]. This observation aligns with the criterion's results, indicating that pair programming contributes to increased efficiency. The positive impact stems from the fact that the method and programming language employed in the project were relatively new to several team members, presenting them with a challenging learning curve. Consequently, pair programming emerges as a valuable approach in such scenarios, delivering favorable outcomes.

5.3 Collaboration

The final criterion for evaluation is collaboration. The objective was to examine the team's progress through the use of pair programming and the time required for the different pairs to synchronize with each other. The findings of this investigation are presented in section 4.3.

In Figure 4, we can observe the development of team members. The graph illustrates the progress of participants on a scale of 1 to 10, where their initial and final levels of expertise were estimated by the respective participants. To gain a comprehensive understanding of their progress, the participants were divided into two distinct groups based on their knowledge level: one with a high level and the other with a low level.

The results depicted in Figure 4 indicate that development has taken place in both groups. Notably, members with a lower knowledge level demonstrated more substantial growth (from 2 out of 10 initially to 8 out of 10 by the end of the project) compared to those with a higher knowledge level (from 7 out of 10 initially to 9 out of 10 by the end of the project). This observation suggests that individuals with greater expertise effectively shared their knowledge, enabling those with less knowledge to learn new concepts, while the more knowledgeable participants refined their existing skills.

[11] `https://www.diva-portal.org/smash/get/diva2:831281/FULLTEXT01.pdf`. K.Nilson

Figure 5 presents the results of synchronization in groups. The project team was comprised of individuals with varying levels of expertise. During pair programming, the team was divided into multiple pairs. Some pairs consisted of individuals with similar knowledge levels, while others had individuals with differing knowledge levels. The purpose was to examine which of these pairs tended to synchronize more quickly.

The diagram in Figure 5, section 4.3, displays the findings. It reveals that pairs with similar knowledge levels required less time for synchronization (approximately 14 minutes) compared to pairs with individuals of different expertise (20 minutes). This can be attributed to the fact that pairs with shared knowledge didn't need to introduce their skills to each other. As they were on the same level, their communication proved highly effective in terms of problem-solving and implementation.

In contrast, pairs with varying knowledge levels needed to familiarize each other with their respective expertise. Each individual made an effort to teach their partner what they didn't know. Consequently, the initial communication was less efficient, resulting in a slightly longer time before both individuals began synchronizing effectively.

In the article by Muhammed T. Haidar and Imran Ali, they delve into the subject of pair programming and unveil its significant advantage: knowledge sharing and transfer[12]. Through their studies, the authors discovered that pair programming outperforms solo programming when it comes to effectively exchanging and disseminating knowledge. This advantageous aspect is further explored in the report, particularly when examining the first criterion known as "Driver/Navigator." The navigator's experience reveals that pair programming fosters a valuable learning environment. Furthermore, the investigation of the final criterion, "Collaboration," highlights a noticeable enhancement in participants' proficiency.

[12]`http://www.diva-portal.org/smash/get/diva2:832682/FULLTEXT01.pdf`. Muhammad T.Haider, I.Ali

6 Conclusion and Future Work

Modern software development has become a central part of society, driving the development of various methods and technologies aimed at facilitating, optimizing, and enhancing quality. One such method that has gained significant attention and usage in the industry is pair programming.

This report delves into the subject of pair programming, conducting an investigation of the method within the context of a software project where it was implemented. The goal of this investigation was to gain a comprehensive understanding of the method's advantages and disadvantages, focusing on three key criteria: Driver/Navigator roles, Efficiency, and Collaboration.

The findings of the investigation were overwhelmingly positive. Drivers reported that working with the pair programming method eased their workload and reduced individual pressure. However, some individuals assuming the Driver role expressed concerns about the potential inefficiencies that could arise from differing knowledge levels between the Driver and Navigator. Navigators, on the other hand, viewed pair programming as a valuable learning experience. Although a few Navigators mentioned instances where they felt their contribution was limited, overall, they recognized the benefits.

The implementation of pair programming in the project demonstrated a significant increase in efficiency. In a substantial portion of the project, the actual time required to complete tasks was considerably shorter than the estimated time, with pair programming outpacing individual programming in terms of speed. However, it should be noted that the method's effectiveness may be reduced when applied to simple tasks, proving most useful for challenging or perceived difficult tasks.

Furthermore, the method fostered increased knowledge levels among all team members, as evidenced by the evaluation of the final criterion. However, pairs consisting of individuals with different knowledge levels required more time for synchronization compared to pairs with similar levels of expertise.

In summary, pair programming has made a consistently positive impact on the work conducted in this project. While a few drawbacks were identified, the overall results highlight the method's effectiveness in enhancing collaboration, efficiency, and knowledge sharing.

For future research, it is recommended to explore pair programming in relation to various aspects, including the influence of individuals' background differences on the effectiveness of this method. Additionally, it is worth investigating the application of pair programming in remote settings, leveraging technology and tools. Comparisons with the traditional approach to pair programming can provide valuable insights.

7 References

A.Cockburn and L.Williams, "The Costs and Benefits of Pair Programming ",from https://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF, (retrieved 28-05-2023).

Yu B.Leau, Wooi K.Loo, Wai Y.Tham and Soo F.Tan, "Software Development Life Cycle AGILE vs Traditional Approaches", 2012 International Conference on Inforamtion and Network technology, from http://ku-fpg.github.io/files/agile-traditional.pdf, (retrieved 28-05-2023).

"Waterfall model" from https://wiki.edunitas.com/IT/en/114-10/Waterfall_22494_eduNitas.html, (retrieved 28-05-2023).

T.Dingsoyr, S.Nerur, V.Balijepally and Nils B.Moe, "A decade of agile methodologies: Towards expalining agilw software development", from https://www.sciencedirect.com/science/article/pii/S0164121212000532, (retrieved 28-05-2023).

A.Srivastava, S.Bhardwaj and S.Saraswat, " SCRUM model for agile methodologu", IEEE 2017, from https://ieeexplore.ieee.org/abstract/document/8229928/authors, (retrieved 28-05-2023).

M.STOCIA, M.MIRCEA and B.CHILIC-MICU, "Software Development: Agile vs.Traditional ", 2013 from http://www.revistaie.ase.ro/content/68/06%20-%20Stoica,%20Mircea,%20Ghilic.pdf, (retrieved 28-05-2023).

"Software development process" from https://en.wikipedia.org/wiki/Software_development_process#:~:text=In%20software%20engineering%2C%20a%20software,development%20life%20cycle%20(SDLC), 1 May 2023, (retrieved 28-05-2023).

Alexander S.Gillis, "Pair Programming" June 2021, from https://www.techtarget.com/searchsoftwarequality/definition/Pair-programming, (retrieved 28-05-2023).

Danielle L.Jones and Scott D.Fleming, "What Use Is a Backseat Driver? A Qualitative Investigation of Pair Programming" IEEE 2013, from https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=6645252&tag=1, (retrieved 28-05-2023).

K.Nilson, "Increasing Quality with Pair Programming- An Investigation of Using Pair Programming as a Quality Tool", 2003, from https://www.diva-portal.org/smash/get/diva2:831281/FULLTEXT01.pdf, (retrieved 28-05-2023).

Muhammad T.Haider and I.Ali, " Evaluation of the Effects of Pair Programming on Performance and Social Practices in Distributed Software Development", June 2011, from http://www.diva-portal.org/smash/get/diva2:

`832682/FULLTEXT01.pdf`, (retrieved 28-05-2023).

S.Faja, ”Evaluating Effectiveness if Pair Programmming as a Teaching Tool in Programming Courses”, November 2014, from `https://files.eric.ed.gov/fulltext/EJ1140923.pdf` (retrieved 28-05-2023).