Pros and Cons Using Mob Programming in Agile Development

1st Katja Palmkvist
Dep. of Computer Science
Lund University
Lund, Sweden
ka1347pa-s@student.lu.se

2nd Filip Myhrén
Dep. of Computer Science
Lund University
Lund, Sweden
fi8057my-s@student.lu.se

Abstract—Programming as a social activity is something that has caught more attention the last couple of years. However, since mob programming, or ”mobbing”, is a rather new phenomena to the software developing world, it have not yet made its way into the academic world. When mob programming a group of three or more people gather within different roles around one screen and keyboard while developing software. This study was done on a group of second year students at Lund University, who had no former mob programming experience. Additionally a person from the industry (Axis Communications in Linköping) was interviewed regarding their experiences with mob programming. The results showed that mob programming comes with a lot of advantages, such as; higher code quality, increased learning, team building and much more. Furthermore some drawbacks or difficulties was found practising mob programming; less code was produced each iteration and the risk of team members losing focus was present. In conclusion it can be seen that mob programming make up a useful tool that suits some situations better than other. The reason that some developers have a hard time trusting in mob programming, is probably because the most considerable benefits from mob programming shows in the long run and is not as visible instantly.

Index Terms—mob programming, agile development, extreme programming

I. INTRODUCTION

When developing software for a nuclear power plant, or perhaps an aircraft, it is crucial that everything works as intended and that there are no bugs jeopardizing the safety. At the same time, programming always comes with a risk, the human factor. Another aspect faced frequently, as a programmer, is complex problems, requiring complex solutions. Solving these problems on your own may take a lot of time, while once you have advised some of your colleagues the solution might appear clearer.

Can making programming a social activity solve the explained issues? Creating software in a group would perhaps both enhance the reliability, but also improve the solving of complex issues. Therefore, this deep study will focus on the application of mob programming in a group of computer science students. Can mob programming help solve some of the issues faced when developing, or are there perhaps more drawbacks with programming as a team than as an individual or in a pair? These are some of the questions this study is aiming to answer.

II. BACKGROUND

This section will explain the background information necessary, in order to comprehend all parts of the study.

A. Mob Programming Explained

The thoughts of mob Programming was started from Woody Zuill, who has become more and more associated with different kinds of group programming [1]. Mob programming is when three or more people write code together with one screen and one keyboard [2]. The size of the screen is important, in order for the mob to be successful everyone have to be able see the code on the screen. There are a few different roles making up the classic mob. The first one, called the driver, holds the keyboard and is writing the actual code. The Navigator has the role of leading the mob and the conversation, he is also the one who tells the driver what code to write. The rest of the members in the mob have the responsibility to be active and to come up with ideas how to solve problems. The roles in the mob rotates in an given interval, which usually measures 10-15 minutes.

B. Programvaruutveckling i grupp (PVG) project

The PVG project is the practical part of a course given at Faculty of Engineering (LTH) at Lund University, where the students learn about developing software as a group. The project is used as a learning tool for the students on how to practise agile software development, and specifically Extreme Programming, in a bigger group (8-10 students). During the project the team spends one full day developing software each week for six weeks. During the lab sessions the students implements stories that the supervisor (also known as the customer) has prioritised beforehand. Beyond the lab session the team has one hour of retrospective with the coaches each week, where they can reflect on the latest lab session.

1) Prerequisites: There are some aspects of the study done that are important to present before going into the results provided. Firstly, the researchers, who are acting as coaches, have no prior experience from mob programming and have not practised the technique themselves before.

The coaches coached two teams during a period of seven weeks. One of the teams were presented to mob programming during their third iteration and continued the practice regularly
during the rest of the project. To introduce the team to mob programming a short presentation about the different aspects and roles of mob programming was held. After the presentation team members where allowed to ask questions and discuss the practice. Finally the plan of how to perform the practice in the given context was presented. The team was divided into two groups, or mobs, of four members each. The iteration time that was used was 20 minutes, excluding git activities. The reason for choosing a longer iteration time was to decrease overhead time, since the students are not yet used to tools like Git.

In the beginning the mob programming was scheduled in the mornings only, however, as the time passed, the team members wished to use the programming technique when it suited their work and were allowed to do so. The composition of members of the mobs differed between each session and every other hour two of the mob members in each mob switched mobs.

Finally, the current situation with an ongoing pandemic (Covid-19) constrained the project to exclusively online work.

C. Axis Communication Team Information

A member from a software development team at Axis Communications in Linköping was interviewed, in order to gather data from the industry and get a wider perspective. The Axis team uses agile methodology and have been using mob programming for over a year. They are a self learned mob programming team, which means that they did not have an external help to start their mob programming project. On the practical part there were a lot of similarities on how the mobbing was done at Axis and in the PVG team. The size of the team in Axis is 6 members and they mob in groups of 3. The iteration time was 8 to 10 minutes.

The two biggest differences in the setup were that the Axis team worked remote on a common company computer instead of their own computer, and that they mobbed every day from 9 in the morning until 3 in the afternoon, with the right to go of and work on your own if that was necessary. Finally, the Axis team worked with a mix of agile and Kanban.

III. RESEARCH QUESTION

- What are the perceived benefits and drawback of mob programming in an agile environment?

IV. METHOD

To gather data during the study four different approaches have been used.

A. Online Survey

After each of the lab sessions, where mob programming had been used, the students answered an online survey. The survey contained specific questions for each session depending on how the mobbing was executed that given session.

B. Informal Interviews

During both the lab session and the reflection hour informal interviews where conducted. These interviews were used to get continuous feedback on the mob programming, to be able to do small adjustment on how the mobbing was implemented.

C. Axis Communications Interview

As mentioned in the background, an interview with an employee from Axis was done, in order to use as a complement to the data gathered from the PVG team. During the interview the data from the online survey was cross referenced to be able to compare similarities and differences from the industry and a student environment.

D. Literature Studies

To deepen the understanding and results, other papers that has been done on mob programming has been read. The results from these studies were compared to the results from the PVG and Axis team.

V. RESULT

In this section the data and results from the study will be presented.

A. PVG Project Data

There were no students in the team that had tried mob programming before. The general opinion of mob programming before the first mob iteration was, as seen in the diagram in figure 1, below, over all sceptical.

![Fig. 1. Students initial attitude towards mob programming.](image)

The main concern of the students was that the mob programming would effect the productivity of the team. However, after the team had tried mob programming for two iterations the general opinion had changed and become over all optimistic. The final attitude regarding mob programming in the team can be seen in figure 2 below. The data was collected after the last iteration in order for the team members to have a wider experience.

![Fig. 2. Final attitude regarding mob programming in the team.](image)
did not follow along as desired. The final result of perceived drawbacks of the team members can be seen in the diagram in figure 4 below. As seen in the figure, none of the students registered the drawback of mob programming being more intense and exhausting.

When asked if they are following the guidelines set up for the mob programming, with roles, iteration time etc., the students answered that the guideline was not always followed. See diagram in figure 5 below.

After the last iteration the students were asked if they would recommend a future PVG team to use mob programming in their project and if they would consider using mob programming in the future, in their education or on a workplace. A clear majority was positive to using mob programming again in the future. See the results in figure 6.

**B. Axis Communications Data**

The most important benefit from doing mob programming on Axis was that the knowledge spread faster throughout the team. This was also beneficial with new recruitments, introducing a new team member went smoother with all team members present, able to answer questions, that appeared during the work, instantly. Other benefits that they experienced was; a higher code quality and that the code became more homogeneous. Additionally, the team got to know each other better, working close to each other every day for a longer period of time.

The drawbacks that the Axis team experienced was that they had to weigh quantity over quality and the fact that it is more intense to mob program and thus there is a risk of people losing focus. Additionally, something that was discussed was the fact that the members of the mob tend to adapt the same conceptions and that after sometime perhaps a new element had to be introduced in order to get multiple perspectives on problems. That mob programming was not suitable for e.g. writing documentation was also mentioned.

Furthermore, it is important that the team members are open for trying this kind of new methodology. All the software development teams at Axis Communications in Linköping have tried mob programming and the team that was interviewed
was the only team which still proceed mob programming daily, during the entire day. It can be established that mob programming is not for everyone and not for every team.

The iteration time and the number of people in each mob was something that had developed during the process of introducing mob programming in the team. In the beginning the team had more frequent retrospectives in order to re-evaluate the mobbing and determine which practices suit the team best.

C. Literature Study Data

In the following section the results from the literature studies are presented.

1) Study 1: The following results originates from study, similar to this study, on a team that used mob programming in the industry [3]. The study found multiple positive effects of mob programming. There were fewer tasks ongoing at the same time, which increased the work flow. The collective code ownership increased and how to write and design code became more consistent. Furthermore, knowledge spread through the group and the understanding of the system increased. Mob programming also had the effect of raising the confidence of the members when they wrote code. The team moral also increased due to mob programming and the time spent on integrating new team member became faster. Finally the team became better at estimating tasks.

There were also negative effects due to mob programming. In order for mob programming to work, everyone in the team have to believe that it will work. The initial production speed can be slowed down due to members of the mob being inexperienced in the current part of the code, which may slow down the whole mob. If members of the team does not get along, their bad relationship can be intensified and harm the team. Switching computer when mob programming in this study gave them a big overhead, which was solved by using one company computer that was set up for just mob programming.

2) Study 2: In this paper the author explains how mob programming can be used to keep the technical dept to a minimum [4]. The conclusion is that if programming is done in a group regardless if it is in a pair or in a mob the understanding of the software architecture will increase and thereby the technical dept will decrease.

3) Study 3: In this article, the use mob programming as a learning tool was experimented with [5]. The experiment used one junior team and one senior team. The junior team adapted to mob programming and it increased their learning curve. The senior team however, did not like how mob programming changed their routines and dropped it after three sessions.

VI. ANALYSIS

In the following section the data from the questionnaires and interviews are analysed and focused into specific advantages and drawbacks perceived of both the students, the Axis team and the literature studies.

A. Pros

When examining mob programming multiple advantages came in to sight. The following section will explain what advantages was found in the study, divided into a few aspects.

1) Higher Code Quality: One of the more obvious advantages of mob programming is the improvement of the code quality it brings. With multiple eyes on the code, there are less risk of making simple errors. Additionally, bad practices and code smells occur more rarely. The mob programming can act as a peer review, meaning that less time has to be spent on reviewing code afterwards. This is something that was detected in the PVG team and confirmed by both Axis and the literature study. The fact that this was found in all of the resources used in the study reinforces this advantage.

2) Homogeneous Code Base: Developing in a team, instead of individually, will result in a more homogeneous code base. This was found both from Axis and in the literature studies. The PVG team did not mention this advantage. This could be a combination of the short time frame for the project and the fact that they were not able to see the benefits of a homogeneous code base in the long run. However, when the team where asked about if they agree with mob programming contributing to homogeneous code, 50 percent of the team agreed with the statement.

3) Advanced Problem Solving: Being in a group of people the brain capacity is inevitably going to increase compared to single or pair programming. The collective intelligence in the mob makes solving more complex problems easier. This was experienced in the PVG team, but not confirmed in the literature study. The developer from Axis did not mention the more advanced problem solving as such, however, he did mention the fact that better solutions were provided with mob programming. It is therefore hard to declare if this effects less experienced programmer more than senior programmers. It can also come from the possibility that this benefit is a truism when it comes to mob programming and is therefore not worth mentioning. To conclude, mobbing undoubtedly comes with better problem solving, in one way or another.

4) Increased Learning: Furthermore, when working together in a mob there are all kind of experience and knowledge. Collaborating and solving problems together is a great opportunity to share knowledge and to learn from each other. This fact has been found in all research done for this paper, in the PVG team, in Axis and in the literature study made. Axis also stated that this benefits as the greatest benefit from doing mob programming. Increased learning can therefore be acknowledged as one of the strong benefits of mob programming.

5) Increased Confidence: The students in the team found mob programming to be a tool to dare writing code and increase confidence in coding. The opportunity to discuss the solution and having other developers watch the process made the team members more confident in the code they where writing. Although the shared code ownership was in place, there existed an increased hesitation changing the code base in pairs than in the mob. This was also found in the
literature study and is therefore not only something that effects programmers at a student level.

6) Better Overview: Working in a mob made the team members gain a better overview of the project. This effect was found in both the PVG team and in the literature studies. This benefit may derive from the fact that team members are present when most of the code is written and therefore does not have that much code which they are not acquainted with. On the other hand, there are more people present when you produce code, so there will always be somebody that can explain the code, if you do not understand a certain part of it. This fact was not described by the Axis team member and not asked either, however, he did mentioned the benefit of all team members being involved when developing, but more with the benefit of everyone being part of, and owning, the solution.

7) Introducing Team Members: One aspect that was never tested in the PVG project but was found both at Axis and in the literature study was the fact that mob programming can be a useful tool when introducing new members to a team. Introducing new team members can be challenging, with a lot of new things to get acquainted to. However, by mob programming the other team members can narrate and stop to explain the parts that needs explanations. What is essential here is that the new member is not afraid of asking questions and being vulnerable, the Axis team member stated.

When asked in the PVG team three out of eight team members agreed with the statement of mob programming easing the process of introducing a new team member. This however, is just a qualified guess from the team members, since there was no possibility for such an introduction in the course project. With the lack of experience in this area, it stands unknown if this benefit can be shown in an academic environment with second year students.

8) Team Building: Practising mob programming demands a certain team productivity. When working in a mob every day together with colleagues it is inevitable to get more close. Having a bad relation with someone that you work as close to as you do in mob programming will be hard and therefore you will have to settle you disagreements. This was detected at both Axis and in the literature studies. When asked to the PVG students six out of eight students agreed with the statement. Since the majority in all of the sources in the study it is clear to see that team building is one of the benefits of mob programming.

B. Cons

The data gathered showed that there was some drawbacks to mob programming. Those will be explained in this section.

1) Decreased productivity: The most clear drawback from the PVG team was that the productivity measured by the amount of code written was decreased. This was an initial feeling that the student had before trying mob programming and they felt that it was a correct assumption. This was not as clear from Axis point of view and not from the literature studies either. All though objective measurements could have been performed on this part, it has not been done in any of the studies in this paper. The PVG project’s short time frame limits the possibility to measure the effects of technical dept, which makes it difficult to compare the results from the practical study done with the statements in the literature. This could also be why this was mostly experienced in the PVG team. Since the other teams in this study have tried mob programming for longer time, perhaps enough time to benefit from a lesser technical dept, which exceeds the lower amount of code produced.

2) Lower Motivation and Inactivity: Mob programming could also lead to low motivation and inactivity. That was especially evident in the PVG team when the mob programming occurred in the afternoon. This was not found at Axis or in the literature studies. This can point to that there are some differences on how mob programming is done in the industry compared to how it is done at a student level.

3) Specific Tasks: All tasks and stories are not suitable for mob programming. Simple tasks, where the way forward was obvious, did not work as well with mob programming. When using mob programming on these sorts of tasks the students felt that their time could be spent better doing separate tasks. This was also confirmed from Axis but not in the literature studies. That does not mean that the literature studies contradict this drawback, it only says that this was never brought up in the studies. The tasks that were mentioned to not be suitable for mob programming according to the PVG team was smaller tasks, documentation and writing tests. Whether all these tasks applies to the industry is not known, however, Axis also stated that documentation of code is one of the tasks less suited for mob programming.

4) Intense: That the intensity while mob programming was higher was not something that the PVG team experienced. However, the Axis team did. This may also indicate that there is some differences on how an experienced team work with mob programming in contrast to a team of students. Whether it is because teams from the industry uses the power of mob programming more and therefore makes it more mentally challenging is hard to say. At Axis Communication they tried to mitigate this by being prompt to take smaller breaks.

5) Not for Everyone: Mob programming is not for everyone. Mobbing is a social activity where you have to be able to compromise and come up with solutions together. This was something that was learned from Axis. If you are not interested of this kind of working method it is not going to work in the long run. This fact was also found in the literature study where one of two teams dismissed mob programming. It should also be mentioned that the PVG team had the opportunity to use mob programming the last lab session and choose not to during most of the lab.

VII. DISCUSSION

In this section the methods used in the study will be discussed and possible related work will be presented.

A. Reflection

The methods used to gather data in this study was of a more subjective manner. The study is built on asking people for their
opinion and thereby also getting a subjective answer. A more objective approach could have been chosen. Tools to measure both quantity and quality of code being produced while the team was mobbing or not could have been used. This way of measuring data also comes with disadvantages, such as the difficulties of measuring the quality of the architecture and code. These difficulties are the reason that a more subjective way was chosen for this study. However, the problem with having to rely only on subjective data is that it is just opinions and feelings. E.g. if a team member think that mob programming is slowing down the working pace, it is that persons reality. The actual reality may be different. The point is that it is hard to confidently say if all of the pros and cons found in this study actually occurred, or if the conception of the team have twisted the reality.

There is also a risk that the students thought that the coaches (as some sort of authority) wanted them to have a positive or negative experience from mob programming. This could have led to either conscious or subconscious bias towards the answers from the surveys.

The amount of sample data is not very large, the team have tried mob programming for a total of three iterations. Some of the experienced benefits and drawbacks of mob programming is something that the team could did not have the opportunity to notice in the short time frame of the PVG project and might have appeared clearer in a larger project with an expanded time frame.

B. Future Work

This study was conducted with the PVG project as focus to get a more general view of the benefits and drawbacks of mob programming. Future work could be perform a more specific study on how mob programming effects the learning in an academic environment. Additional future work could also be to conduct a similar study but perform objective measurements to see the objective benefits and drawbacks.

VIII. Conclusion

When examining the use of mob programming it has been discovered that the motivation of the team to use mob programming is not always high. In this study the initial attitude against mob programming was something that changed after trying the method in practice. Mob programming was found to have multiple benefits, which, in the long run, can have a essential impact on a team’s success. One of the most considerable benefits being increased learning. Additionally, mob programming does not come without drawbacks, for example, normally, less code is produced each iteration. Finally, mob programming can be used to increase both team spirit, code quality och advanced problem solving. Mob programming is not for everyone and not for every task, however, trying mob programming with your team may give you great improvements and take your software development to new heights.

ACKNOWLEDGEMENT

We would like to express our gratitude to Yin Fai Chan for our conversation about mob programming at Axis Communications in Linköping.

REFERENCES


