EXPLORING THE IMPACT OF MOB PROGRAMMING ON THE WELL-BEING OF DEVELOPERS: INSIGHTS FROM A SOFTWARE COMPANY

Philip Björklund
Department of Informatics, Linnaeus University, Sweden
pb222np@student.lnu.se

Jacob Fridebo
Department of Informatics, Linnaeus University, Sweden
jf222vt@student.lnu.se

Fisnik Dalipi
Department of Informatics, Linnaeus University, Sweden
fisnik.dalipi@lnu.se

ABSTRACT

Research has demonstrated that software engineering teams nowadays face numerous challenges that revolve around the well-being (happiness) of the developers. One way to address these challenges can be through Mob Programming (MP). Mob programming represents a novel software development practice where a whole team works together on the same coding problem, at the same time, in the same space and on the same computer. In this study, we investigate the impact of using MP on the well-being of software developers. A qualitative method with semi-structured interviews and observations was applied for its ability to extract in-depth and valuable information. The participants selected for this study derived from four different development teams who worked at Fortnox AB in Växjö, where one of the four development teams used MP on a daily basis and the three other teams practise MP on an occasional basis. A total of 13 interviews were conducted at the company's offices. Thematic analysis is used to organize and create a structure regarding the information from the interviews. Two themes with specific sub-codes are created based on the thematic analysis: Team dynamics and Individual dynamics, which are derived from the interview questionnaire. The study found that the majority of software developers were impacted in a positive way regarding well-being while practising MP. Reduced stress and individual work pressure, and increased social interaction were two of some prominent factors that contributed to this result. However, stress was also found to have negative effects on some developers, which was associated with the constant attention and overlooking by others during the coding process.

Keywords: *Mob Programming, Agile development, Well-being, Individual Performance, Team dynamics*

1. INTRODUCTION

Studies in industrial and organizational psychology have suggested that understanding happiness and unhappiness could lead to cost-effective ways of improving working conditions, job performance, and to limiting the frequency of psychological disorders (Graziotin et al., 2018). The happiness of all stakeholders involved in software development is an indispensable element of company success (Denning et al., 2012). Software development teams, in particular, nowadays face various challenges which revolve around lack of communication and well-being (happiness) of the developers, and increasing customer demands, time limitations, the quality of the code, among others (Casale et al., 2016; Mistrik et al., 2010; Graziotin et al., 2018).

One way to address these challenges can be through Mob Programming (MP), which is enforcing deep collaboration within software development teams and is leveraging distributed knowledge when solving a problem. MP represents a software development practice where the entire team works on the same problem, at the same time, in the same space, and on the same computer. This agile coding approach extends the concept of pair programming from two people working together to the entire team. Developers in MP jointly work with one team member coding on the keyboard called Driver and the rest of the team which are called Navigators, are observing the Driver to review the code for syntactical or logical errors. According to (Buchan and Pearl, 2018), the most effective team size consists of three to four developers. In such MP settings, participants show more consistent results and enhanced ability to code and design. Other benefits of MP include strengthened learning for the participants, regardless of their background or skills and less distractions and interruptions because the participants are able to choose their environment (Balijepally, 2017).

Nevertheless, due to the very recent proliferation of MP in the universe of software development methods, to this day, many of its advantages and risks are still unclear and have not been thoroughly and widely explored by the research community. The most difficult part when researching about MP deals with data collection and results from MP sessions (Shiraishi et al, 2019). Moreover, due to the fact that such sessions can last a few months to be completed, it is very time consuming.

One of these under researched aspects of MP is also well-being or the happiness among developers while "mobbing". Thus, inspired by the fact that there is inadequate evidence in scholarly studies that have attempted to shed more light on this topic, this paper may contribute to fill this gap.

Hence, the main goal of this paper is to assess the impact of MP on a daily or occasional basis regarding the software developers' well-being. In this context, rather than investigating the entire company or division, the focus is placed on examining the well-being at individual level. As part of this evaluation, we identify that most of the MP practitioners included in the study were positively impacted when it comes to well-being while "mobbing", while there were some other developers who also reacted negatively towards MP as a development method in general.

The remainder of this paper is structured as follows: in the upcoming section we provide a background information where we highlight relevant studies from the state of the art. We provide then an overview of the research design employed in this study. In section 4 we highlight the identified findings, whereas section 5 provides an analysis and discussion based on the findings. Section 6 concludes the paper with some insights for future research directions.

2. BACKGROUND AND RELATED WORK

Mob programming can be seen as an evolution of the pair programming method used in Extreme Programming (Zuill 2014). The basis of Extreme Programming also known as XP is to work together in pairs, having two programmers work together on one computer. The idea behind pair programming is that two brains and four eyes are better than one brain and two eyes. Teams that have used pair programming have resulted in an increased quality of code and work pace. The method also contributed to a more streamlined code which resulted in overall less code ("Extreme Programming", n.d.).

Mob Programming also takes the communication and the whole team approach from Extreme Programming (Zuill 2014). Valuing that face-to-face is the best communication form and having an open space for the team without barriers is key to its success. Team functionally is highly valued in both Extreme Programming and Mob Programming.

At first glance, mobbing appears to be a waste of resources and time when a whole team is working on the same task instead of everyone working independently on different tasks. However, there are a few potential benefits to this arrangement. The first one is the reduced managerial overhead. This means that the employees in a team using Mob Programming are talking to each other to a greater extent and therefore have a much lesser need e.g. emails, meetings, code reviews, and code mergers. This is possible because of the constant teamwork. Every line of code is created by the team and is therefore analysed and discussed by the present members of the development team (Balijepally et al. 2017).

When mobbing, the interpersonal interactions are intense and frequent. This can impact the team positively or negatively depending on the relationships within the group. Any kind of challenge needs to be resolved quickly in an effective way to counteract negative barriers that can affect the team. Since MP is a team-based work method, it tends to reveal if members prefer to work alone and can, therefore, be preserved as non-team players. Communication is also a vital element when it comes to MP, which demands the members to be more open and talkative with each other. This can become a problem if members have different interpersonal skills. The members who are lacking good interpersonal skills may feel isolated and left out (Buchan et al, 2018).

According to (Graziotin et al. 2018) there has been a surge of new research on happiness and its impact on software developers over the last few years. Many studies have tried to exemplify the relationship between happiness and performance. Based on Colomo-Palacios and Casado-Lumbreras's (2011) study, high activation and low pleasure levels were the results of high versioning requirements. Activation decreased over time, while pleasure increased with each iteration.

Based on Wrobel's (2013) study with 49 developers participating showed that while experiencing positive emotions, productivity increased and was impacted positively. The study also showed that while experiencing emotions of frustration, productivity deteriorated and was impacted negatively. Frustration as an emotion in the context of software development has been studied in another survey by Ford et al. (2015) where 67% of the 45 participants felt that frustration was a severe issue. Some of the catalysts that triggered frustration were believed to be related to fear of failure, personal issues, lack of time, and grudges against peers.

According to (Graziotin et al. 2018), happiness and unhappiness have the ability to impact productivity and quality in software development. This will often be expressed by the level of creativity, flow, process-related performance and cognitive performance. Happiness will often benefit the individual developer while unhappiness can affect and be detrimental to others e.g team members.
The research done on happiness in the context of software development shows that it has a deep connection and can impact performance and quality in many different ways. In summary, a lot of present research points towards that affective experiences and happiness are connected with each other and play a big part in performance and quality regarding software development. However, there are still many unanswered questions regarding the subject. The understanding of software developers and their experiences is still incomplete (Graziotin et al, 2018).

2.1 Definition of well-being

The research question chosen for this study revolves around well-being. In this study, the intention is to evaluate the variable with these concepts to give a better understanding of how the employees are affected by Mob Programming. Figure 1 shows the different well-being concepts which are considered in this work.

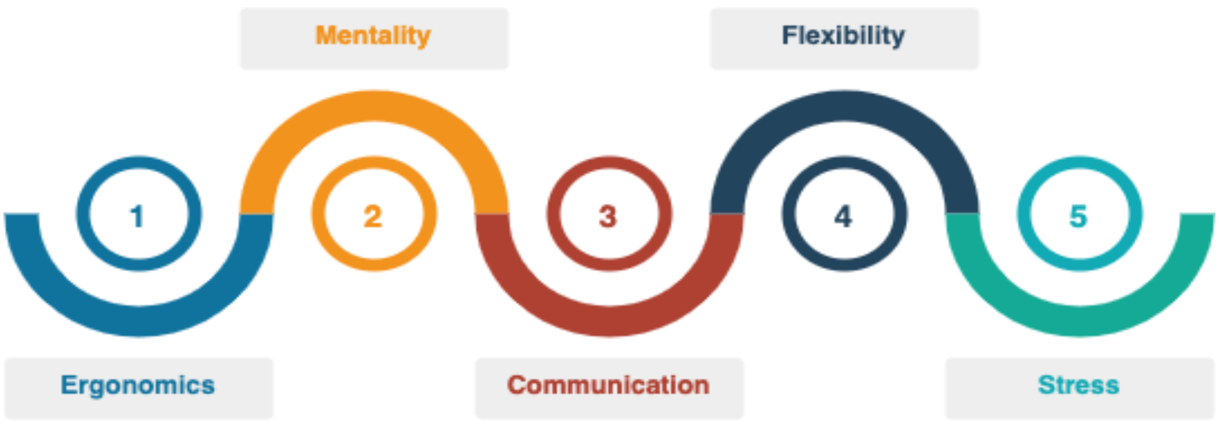

Fig. 1 Different elements of well-being

Below we explain the different elements of employee well-being that we are studying. It is worth mentioning that ergonomics is not included in this section due to the lack of relevant information that we got from the qualitative study.

- Mentality - is believed to be mutable according to Daun (1996). The mentality can also change in correlation with society and other outside influences. However, because the mentality is transferred through generations in the form of upbringing, it is believed to guarantee a historical continuity. In our study, the mentality will be referring to the attitude and mindset of the individuals towards work and their ability to provide results within a mob.

- Communication (Social aspect) - can be defined as the behaviour that enables the sharing of information between interacting entities (Scott-Phillips 2008). In our study, the concept is referring to the ability to build work relationships with team members to have a positive work atmosphere within the mob.

- Flexibility - concerning work is often related to time and location elements according to Gibson (2003). In our study the concept is referring to individual flexibility as a team member in a mob, the possibility to, for example, individually work, use sick days, go to individual meetings, or take breaks.

- Stress - is often linked to the emotional and physiological reaction to stressors which can be circumstances that initiate the stress response (Lloyd et al. 2002). In our study, stress is referring to the level of stress an individual experiences from work and the pressure from workload, or deadlines when working in a mob.

3. RESEARCH DESIGN

To extract in-depth and valuable information for this research a qualitative method was selected. We decided to focus on specific individuals within different teams that possess knowledge and experience of using Mob Programming to understand their perception of the subject. Through this method, we could accumulate relevant information regarding the impact of Mob Program usage.

When it comes to the understanding of an individual's perception of reality, qualitative methods emphasize closeness as the main element. To examine and investigate social phenomenon's we need to understand how humans perceive social reality. Observing the individuals, what they do, what they say, and let them speak by their own words is crucial (Jacobsen 2002). By conducting semi-structured interviews, we could achieve this in an ideal way.

3.1 Qualitative questionnaire and interviews

We created a questionnaire form to define specific questions that could potentially acquire information about the research topic. The questions in the questionnaire were divided into two sections. The purpose of the first section with questions was about getting the interviewee in a comfortable state and warmed up by asking generally easy questions about their work environment and background. The second section of the questionnaire contained the questions directed at the well-being aspect, gathering information regarding the interviewees' experiences and thoughts while using Mob Programming.

The interviews took place in a closed setting, meaning one individual at a time from the specific mob answered the questionnaire. Semi-structured interviews were selected in this case to enhance the data analysis of the information given by the interviewees. Therefore, it was easier to compare and analyse different answers given to the same questions.

In order for the results of the qualitative study to be applicable to more than just Fortnox development teams, focus was given to ensure that the results were general to all users. Using and performing interviews and observations on employees from different teams gave a more general understanding of the subject that could hopefully be applied to different developers and generate similar results. The questions asked in the interviews were carefully analysed beforehand to ensure that they were impartially phrased and gave the interviewees an unbiased basis when answering.

The participants chosen for the interviews derive from a minority population after the limitations were applied to Fortnox's development teams. A total of thirteen interviews were carried out at the Fortnox office. Interviews were carried out on four members in the only development team that used Mob Programming on a daily basis and as their primary work method. A total of nine different interviews took place in relation to the other three teams that used Mob Programming on an occasional basis. This contributed to further enhance the comprehensive experience from the interviewees. Due to the conditions in the letter of consent we have chosen not to disclose information about the participants gender, age and profiles in order to uphold their anonymity. Swedish language was used during the interviews since the vast majority of the interviewees' native language was Swedish. This made it possible for individuals to express themselves in the best way possible. The participants' experience working with Mob programming ranged from 6 to 3 months, where the majority of participants had 6 months experience.

3.2 Thematic analysis

A thematic analysis was used to organize and create a structure regarding the information from the interviews. This process made the information easier to handle and view as we went on to analysing the information.

The interviews were audio-recorded which allowed us to transcribe the interviews word by word to give us a detailed description of the conversations. As suggested by Jacobsen (2002), researchers need to acquire a grounded and detailed rich description of the accumulated data and information from the qualitative study. Situations, interviews, and conversations should be registered as detailed as possible. This process is called "thick descriptions" and often contains plenty of details, analysis, and variations.

In every process of analysing information, there is always a phase that includes screening and simplification of the accumulated information. This is a critical part of the analysis process to get an overview of the information. The "thick descriptions" are comprehensive and therefore impenetrable to others except for the researchers. Systematisation is therefore necessary to be able to convey what has been found in terms of information (Jacobsen 2002). Through systemization, interesting topics and information could be found to make the screening process smoother and more precise.

Through several iterations, we looked through the information after patterns and similarities between the interviews. These correlations were then used to create sub-codes that allowed us to gather repeated answers into information about the interview's experiences. Braun et al. (2006) emphasize that it is important to give each data item the same level of attention to easier recognize interesting aspects that may form repeating patterns. These sub-codes were later categorized into themes that consisted of several sub-codes that had a connection with each other.

Themes capture valuable and important data concerning the research topic. The themes are structured in a way that represents a patterned meaning within the data set from the qualitative study. It is important to review and refine in this process to identify themes that do not fit or blend into each other. This is important to capture and identify the essence of what each theme is about and what aspect of the data they represent (Braun et al, 2006).

4. FINDINGS

This section will introduce the findings and empirical information from the qualitative study. The information was divided into different Themes with Sub-Codes to make the findings more structured. Each of them is presented with its corresponding Sub-Codes, as can be seen in Table 1 below.

Theme	Sub-Code
Team dynamics	Responsibility Interest collision Social interaction
Individual dynamics	Stress Motivation

Table 1. Themes with related Sub-Codes

4.1 Team dynamics

Team dynamics highlight the patterns of interaction among software development team members that determine the performance of the team (Dorairaj et al, 2012). Three variables of team dynamics are identified and studied in this work, i.e. responsibility, interest collision, and social interaction.

4.1.1 Responsibility

The staff in agile teams are often involved with a considerable amount of responsibility according to Coram et al. (2005). A mentor or coach leadership can be a good approach to make teams more effective. It is also important for the team leaders to act in a way that enables the members to take initiative and responsibility. This is done through collaboration rather than command and control. Responsibility is important but an under-researched area when it comes to MP in specific. This study will therefore include it to shed light on it and contribute to the subject. Based on our findings, several interviewees explained that working in a mob means sharing the workload and doing all tasks together which affects the way responsibility is dealt with within the team. One interviewee explained the responsibility aspect of Mob Programming as:

> *"It's pretty nice because you no longer have the sole responsibility for something. Before we used Mob Programming, I always had the responsibility to help other people with issues that I was involved in. This led to a lot of interruptions which affected my work negatively. However, when we use Mob Programming everyone is involved and takes equal responsibility. This also means that everyone can help solve problems that you have been involved in. You can at least point to one or two people that excelled and contributed a lot to the task."*

Another interviewee expressed their opinion on responsibility and what possible negative effects it could have for a team:

> *"I think the responsibility is more collectively, therefore you take more responsibility towards the team than before. But it's also because it's fun and usually it comes naturally. One of the dangers of Mob Programming is that you can have people just sitting passively. It doesn't feel like we've had that, everyone is contributing. Those who feel they are not contributing have done something else instead."*

4.1.2 Interest collision

Conflicts and interest collisions are inevitable in software development according to Zhang et al. (2007). This often occurs due to the members' task and individual differences. It is very important to understand and accept that interest collisions do happen. Inefficiencies can quickly rise if the conflicts are left untouched. This study will further explore the area due to its capability of crippling MP teams and make them inefficient.

In our study, many interviewees explained that Mob Programming is a work method that brought the team together and demanded more communicative teamwork from the individuals e.g. how an issue was going to be solved. Below are a few takes on the topic from different interviewees.

This interviewee highlights that interest collisions do happen sometimes but there are models and work methods that can solve those types of situations:

> *"The closest conflict we have had is where we cannot agree on how to solve an issue. But there is a good model for that, we tested both solutions and decided what to do after that."*

Another interviewee thought that Mob Programming helped and made the process more efficient when interest collisions do happen:

> *"It feels like people have an easier time expressing themselves. So even if there is an interesting clash it has been easier to handle it in that group because you have more opinions on it straight away."*

4.1.3 Social Interaction

Communication and social interactions are important key factors to MP's success. It is important to note that social interaction can be foreign to some individuals in the context of software development. Therefore, it is very important to practice and collaborate within the team to achieve a good social standard Zuill (2014).

Based on our study, the majority of the interviewees agreed that communication is key when it comes to Mob Programming. They believe that the members of the mob must be more social and communicative in their way of working since the behaviour can impact the social interaction between the team members.

This interviewee talked about how Mob Programming made it easier for newcomers to socialize with the other members of the team and sped up the onboarding process:

> *"We take regular breaks where we can chit chat with each other and it helps to get to know each other, so I appreciate it as a newcomer. I think it is a great activity to speed up the onboarding process in the team."*

This citation from another interviewee brought up the social impact of Mob Programming in the way of communicating with members that they usually do not talk to that much:

> *"you talk to the people you don't usually talk to in the group. So, it became like a good relationship-building thing for us, so now we have an easier time communicating than before."*

4.2 Individual dynamics

4.2.1 Stress

Agile software development has many tools and methods to prevent software developers from feeling stressed according to Meier et al. (2018). However, if teams are not managed in a proper way i.e lack of control, high task requirements, or poor interaction within the team, stress can rise up and quickly cause issues. The result from our study revealed that stress concerning Mob Programming can be affected both positively and negatively depending on the individual.

This citation from an interviewee brought up a positive impact on stress about Mob Programming:

> *"I do not feel the same pressure to perform when I am mob programming as I do when I am sitting alone. It feels like I can sit with an issue for several days and feel like I should be done with it at that point, which stresses me out. It's quite the opposite while we are mobbing, everyone knows that it will take a certain amount of time for the specific issue and therefore it will not be quite the same stress for the tasks we do together as a mob."*

Another interviewee pointed out the constant overlooking of what you are doing as a Driver in the mob:

> *"The concept means that everyone is watching what you do. If you sit by yourself, you can make mistakes or do lots of tests without people looking. If everyone is looking at what you are doing, you want to be able to be quick and find everything the Navigators are pointing out which can be hard if you're new to the team."*

4.2.2 Motivation

According to Asproni (2004), there is a strong correlation between effective teamwork and motivation. To increase motivation in agile development, there must be a clear and concrete goal. Competence also plays a big role in motivation as it determines if the goal is attainable or not. Based on our findings, Interviewees pointed out that the motivation towards work is affected by using Mob Programming. The ability within the team to motivate each other was something that stood out from the interviews as a key argument as it was a reliable source of motivation.

An interviewee described their relation to motivation with Mob Programming as:

> *"I feel that the motivation gets better because, if you need a 10 min break or something, you can do it and there are still other members who continue to work. So, there is always someone who is motivated and who 21 motivates others. Therefore, it is easier to be motivated when you need to help others than when you sit at the computer by yourself. So, I think it helps a lot that way."*

One interviewee referred to the flexibility within the mob regarding the continuous work done by the team.

> *"It's quite common that I need to go away and do other things, and then it is very easy to get into the task again when you are using Mob Programming. You also get a feeling that you are contributing even if you can't be with the team all the time."*

5. ANALYSIS AND DISCUSSION

When the different teams changed to work with Mob Programming, the responsibility naturally shifted towards a collective responsibility. Therefore, a specific person no longer held sole responsibility for a specific issue. The interviewees credited this shift to the increased rate in shared knowledge and work understanding through developing and working together as a team. Balijepally et al. (2017) discussed the impact of sharing the workload and the pressure to get it done, arguing that Mob Programming reduces the individual pressure and workload. The interviewees in our study expressed that this was a positive feature of using Mob Programming and brought benefits both on a team and individual level.

Interest collisions are a common occurrence when working together as a team and having everyone be a part of the same development process. However, the result from our interviews pointed towards Mob Programming solving the issues easier and more efficiently. The interviewees explain that they felt like people had an easier time to express themselves in the team, instead of online or through a Pull Request. Usually, the interest collisions consisted of different ways to solve the issue where the team could not agree on a specific solution. A good model for solving this used by the teams was to test both solutions and decide what to do afterwards. This made the solutions more bulletproof and helped the team decide what to use. Balijepally et al. (2017) expressed benefits to communication when using Mob Programming as it simplifies the communication flow within the team and allows versatile discussion supported by knowledge from all team members.

Because of the importance of having good communication within a mob to be effective, social interaction became an important aspect for the teams. Getting to know each other and build a good work relationship made the groups more effective. The findings from the interviews were that Mob Programming helps further enhance this process and helps the team members build a working relationship with each other. The mob process helped them interact with all team members and gave them the possibility to socialize with their co-workers in a better way. The interviewees also expressed that Mob Programming made them interact more with people they usually did not talk to, building relationships that made it easier to communicate in the team than before.

The result regarding stress showed that there is a mixture of both positive and negative impacts on the theme when using Mob Programming. The qualitative study showed that the impact on stress was often based on the individual in question. Balijepally et al. (2017) mention the positive impact Mob Programming can have when it comes to stress. Since Mob Programming was all about working together and helping each other, there was a reduced amount of individual workload. It also reduced the constant pressure to get things done all by yourself. The structure of Mob Programming was designed to be used as a team where helping others reward the general process. However, for some people, this structure could impact their stress levels the other way around. As a result of the qualitative study mentions, some individuals felt like they were constantly being watched, overlooked, or judged by their work. This resulted in an uncomfortable feeling which could be hard to deal with. Therefore, it could be hard for some individuals to get used to the ways of Mob Programming.

The motivation was shown to be impacted similarly due to its correlation with stress. The individuals who felt less stressed when using Mob Programming tended to get increased motivation and incentives to put in work. On the other hand, people who felt more stressed when using Mob Programming tended to get reduced motivation and fewer incentives to work hard. Buchan et al. (2018) also emphasized that motivation could be affected depending on the individual's coding experience. Individuals who had an inferior coding experience concerning the person's team members tended to become a passive spectator instead of actively participating in the mob. This could result in reduced motivation when using Mob Programming for some individuals.

The existing research found regarding the handling of responsibility when using Mob Programming was very minimal in terms of its effects on the team members. The information that could be reinforced through the findings of the study was the theory Balijepally et al. (2017) discussed arguing that Mob Programming reduced the individual pressure and workload. The findings from our study suggested that collective responsibility also reduced individual pressure and workload. We believed the change in perception of responsibility when using Mob programming was related to focusing less on individuality and more towards working together as a mob. A product of that was the new form of responsibility developed when using Mob Programming, a responsibility to contribute and perform within the mob. Some of our interviewees expressed that the new form of responsibility made them more motivated to work. We believed that this was derived from the combination of social pressure within the team to perform and the sharing of new knowledge within the team.

The findings regarding interest collision, the occurrence of it, and how to handle it, showed a positive impact on the usage of Mob Programming. The result implied that the Mob Programming structure was effective in producing an open environment for discussion between team members and the opportunity to further improve communication. This confirmed the communicational benefit Balijepally et al. (2017) expressed with Mob Programming. By having the whole team working together, a more versatile discussion was possible, and the team could use their various types of knowledge to further analyse the solutions from different perspectives

The social aspect of using Mob Programming was an interesting topic and from the interviews, the result showed an increase in work relationship building as well as interaction with co-workers. The existing research around social interaction with Mob Programming was very limited but in the case study by Buchan et al. (2018), the result was that Mob Programming helped the onboarding process of new members, which could be reinforced by the interview results. We believed that Mob Programming helped the team get to know each other better and created an environment that was open and flexible for the intent of socialization. We valued the importance of good communication and believed that to perform as effectively as possible in a team, knowing your co-workers well would bring large benefits to the team.

The findings in our study regarding stress were very scattered and separated in comparison to the findings of existing research work. The majority of the interviewees in our study pointed out positive aspects regarding stress which reinforced a big part of the existing research work done on Mob Programming. However, some interviewees were pointing out negative effects regarding stress when switching to Mob Programming e.g. constantly being watched. We believed these mixed findings were natural since individuals tended to react in different ways when they worked closely together. These findings give interesting insight when analysed since it pointed out both positive and, in some cases, negative aspects.

Results found regarding motivation correlated a lot with the stress aspect. If the interviewee felt less stress when using Mob Programming the motivation increased. On the other hand, if the interviewee felt more stressed when using Mob Programming the motivation decreased since the individual did not feel comfortable with the work method. Buchan et al. (2018) mentioned that motivation often correlated with the individual's coding experience. If the person had inferior coding experience concerning the other team members, the person tended to become more of a passive member of the team. The existing findings did not correspond at all with our findings from the qualitative study. There were several individuals with inferior coding skills that showed a great amount of motivation and willingness to learn when using Mob Programming. We believed that coding experience impacted motivation less than stress. Our findings strongly pointed to a correlation between stress and motivation and helped us understand why people felt less or more inclined to use the work method as well as what variables affected their thought process.

6. CONCLUSION AND FUTURE WORK

In this paper, we have addressed the impact of MP on software developers' well-being. It highlighted many important challenges developers face today when it comes to practising MP. Additionally, this work provides several interesting findings in this direction.

We used a qualitative thematic analysis to investigate *responsibility, interest collision, social interaction, stress, and motivation,* which are sub-codes belonging to team and individual dynamics themes, respectively. Our investigation yielded that a considerable number of interviewees are positive towards MP, claiming that it reduces individual pressure and workload, promotes open environment for discussions and brainstorming, contributes to communication improvement, and enhances social interaction. However, we found that stress was the only variable that brings negative effects to some individuals while mobbing, largely contributing to decreased motivation. The main reason behind stress creation is the feeling of being constantly the centre of attention, or constant overlooking.

For future work, we plan to extend the number of observations, increase the study sample size, and conduct a wider investigation across different organizations. This will allow us to undertake a more comprehensive evaluation on the impact of MP on developer's well-being or happiness and confirm the validity of the findings and challenges identified in this paper.

7. ACKNOWLEDGEMENTS

We would like to thank Fortnox AB for the given opportunity to perform our study at their premises, and the 13 software developers from the same company who participated in the interviews. This study is based on the degree project of the first two authors.

REFERENCES

Asproni, G. (2004) "Motivation, Teamwork, and Agile Development." *Agile Times,* **Vol. 4**.

Balijepally, V. Mahapatra, R. and Nerur S. (2006) "Assessing Personality Profiles of Software Developers in Agile Development Teams" *Communications of the Association for Information Systems,* **Vol. 18**, pp. 55-75.

Balijepally, V. Nerur, S. and Chaudhry, S. (2017) *Mob Programming - A Promising Innovation in the Agile Toolkit.* University of Texas at Arlington.

Braun, V. and Clarke, V. (2006) "Using thematic analysis in psychology", *Qualitative Research in Psychology*, **Vol. 3**, pp. 77–101.

Buchan, J. and Pearl, M. (2018) *Leveraging the Mob Mentality: An Experience Report on Mob Programming*. Christchurch, New Zealand: Association of Computing Machinery.

Casale, G. Chesta, C. Deussen, P. Di Nitto, E. Gouvas, G. Koussouris, S. Stankovski, V. Symeonidis, A. Vlassiou, V. Zafeiropoulos, Z. and Zhao, Z. (2016) “Current and Future Challenges of Software Engineering for Service and Applications”. *Procedia Computer Science,* **Vol. 97***,* pp. 34-42.

Coram, M. and Bohner, S. (2015) "The impact of agile methods on software project management," *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, Greenbelt, MD, USA, 2005, pp. 363-370.

Colomo-Palacios, R. Casado-Lumbreras, C. Soto-Acosta, P. and Garcia-Crespo, A. (2011) “Using the Affect Grid to Measure Emotions in Software Requirements Engineering” *Journal of Universal Computer Science*, **Vol. 17**, issue 9, pp. 1281-1298.

Denning, P.J. (2012) “Moods” *Communications of ACM*, **Vol. 55**, issue 12.

Dorairaj, S.,Noble, J. and Malik, P. (2012) “Understanding Team Dynamics in Distributed Agile Software Development.” *Agile Processes in Software Engineering and Extreme Programming,* **Vol 111**. pp. 47-61.

Extreme Programming. (n.d). Retrieved from https://www.agilealliance.org/glossary/xp

Ford, D. and Parnin, C. (2015) Exploring Causes of Frustration for Software Developers, *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, Florence, pp. 115-116.

Graziotin, D. Fagerholm, F. Wang, X. and Abrahamsson, P. (2018) “What happens when software developers are (un)happy”. *Journal of Systems and Software,* **Vol. 140**, pp. 32-47.

Jacobsen, D I 2002, *Vad, Hur och Varför?*, Studentlitteratur AB, Lund

Meier, A. Kropp, M. Anslow, C. and Biddle, R. (2018) “Stress in Agile Software Development: Practices and Outcomes.” *Agile Processes in Software Engineering and Extreme Programming,* **Vol. 314**. pp. 259-266

Mistrík I. Grundy J. van der Hoek A. and Whitehead J. (2010) *Collaborative Software Engineering: Challenges and Prospects.* Springer, Berlin.

Shiraishi, M. Washizaki, H. Fukazawa Y. and Yoder, J. (2019) Mob Programming: A Systematic Literature Review, *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC),* pp 616-621.

Wrobel, M. R. (2013) Emotions in the software development process. *6th International Conference on Human System Interactions (HSI)*, Sopot, pp. 518-523

Zhang, X. Dhaliwal, J. and Gillenson, M. (2007) “Interpersonal Conflict between Developers and Testers in Software Development” *Department of Management Information Systems Fogelman College of Business and Economics University of Memphis, Memphis, TN 38152*

Zuill, W. (2014) *Mob Programming: A Whole Team Approach.* Application Development Manager at Hunter Industries, Inc., Agile Coach.

Zuill, W. and Meadows, K. (2016) *Mob Programming: A Whole Team Approach*. Leanpub.