TECHNISCHE UNIVERSITÄT
CHEMNITZ

DeVLoger lives: A qualitative approach to investigate different aspects of software developer teamwork based on VLogs data

Master Thesis

Dept. of Computer Science
Chair of Software Engineering

Course of Studies
WEB ENGINEERING

Submitted by: Akash Ashok Chougale

Date: 04.09.2022

Professor: Prof. Dr.-Ing. Janet Siegmund
Supervisor: MA. Elisa Madeleine Hartmann

Abstract

Software development is a lucrative and popular career option, and incites curiosity even in the general public unfamiliar with programming. For software developers teamwork is one of the important part of work. This research shed a light on the topic of teamwork in the context of software development. The objectives are to identify the factors affecting on software developers while working in the team. To find out the it I analyzed 30 vlogs by software developers on YouTube and conducted personal interviews with 5 software developers. I discovered many issues that software developers experience while working in teams, including communication within the team, availability/dependency, interruption, and meetings. Also we found that, more meetings and interruptions can make negative impact on software developers's productivity and efficiency.

Keywords: Teamwork, Meeting, Pair Programming, Interruption, Remote work

Contents

List of Figures

List of Tables

1. Introduction

This chapter introduces the background of the subject “A qualitative approach to investigate different aspects that have a significant effect on software developers while working in a team based on vlog data” and hence the motivation for this thesis. I tried to find existing research in this context to find out the factors that affect software developers while working in a team. I found very less material that focuses only on software developers as most of the research considered the whole team. To do more research in this field, I collected the data by using vlogs and personal interviews with software developers. I analyzed the collected data and selected a few categories where software developers spend more time and face issues in their daily routines. Based on these categories, I designed an interview questionnaire. Based on that, I will discuss and conclude the results of our research.

1.1. Background for the thesis

Software development is a highly lucrative and popular career choice that piques the interest of even those who are unfamiliar with programming. Vlogs (video blogs) are becoming popular among software engineers to explain what it’s like to walk in their shoes daily. Developers give a broad view of their technical work, covering programming sessions, team meetings, as well as their personal lives through these vlogs. Vlogs by software developers, on the other hand, give a personal look at many jobs in the software development sector outside of technical labor. This vlog's transparency offers a larger view of what developers do in their working days, how they interact with their coworkers and the factors that influence their daily performance while working in a team.

Working in a team is one of the main parts of the job of a software developer. The number of technologies used today also means that it is difficult for one person to know enough to do everything required for putting together a professional application. Many applications are split well into front-end and back-end, with people very skilled at each but not nearly as skilled at both. It can be difficult for people to switch from writing a UI and then to tuning a database and then switching to tracing lost packets.

Teamwork is important in software engineering. Teamwork has been extensively studied in several fields, but little of this knowledge has been applied in the context of software development. In the most popular agile method, Scrum, work is organized into small, cross-functional teams with a facilitator and team members. Team members coordinate their work frequently, such as in daily stand-up meetings [18]. Vinekar et al. [17] explain that agile development and traditional development have different views on teamwork. One of the reasons may be that general knowledge needs to be tailored to software development to become useful. As a result, there is a need for further research on teamwork in this specific field.

Extreme Programming (XP) and Scrum are agile methods that direct software development in small, self-managing teams. While there are reports of major improvements with agile development methods over traditional development methods [30], effective teamwork is still a challenge. In several studies in small and large Scrum teams in various consulting and product development settings in companies of variable sizes over the last five years, I have observed three recurring challenges:

Communication: Communication throughout the lifecycle of a project is one of the biggest difficulties in a software developer's life. Bad communication in development teams can occur when the team leader does not have a clear picture of all activities, brief emails that lack clear guidelines, extended meetings that might have been emailed; and too many distractions from coworkers when working on a tough task. These sorts of scenarios cause information flow disruptions and personal confrontations between developers and team leaders. In today's environment, where teams are dispersed across time zones, it requires a lot of effort to coordinate all the tasks among various developers. It is difficult to comprehend the challenges that certain team members are experiencing. Important decisions are sometimes made by the project manager without consulting the team. This occurs despite agile development methodologies' significant emphasis on communication and collaborative decision-making. In the daily meeting, however, many team members approach only the team leader and not the entire team. Furthermore, I noticed team members who did not pay attention when critical issues were presented, as well as a few who fell asleep in meetings. In our survey, I am discussing with participants their challenges in communicating with team members.

Availability / Dependency: This problem is deeply related to communication. After the COVID pandemic, companies allowed their employees to work from home or remotely. People who work remotely must balance their professional and personal lives. For instance, if they are working from home with a newborn, they may not always be there at the same time as others. They might begin and end at different times, not to mention leave their "workstation" during the day. Even if they give a detailed timetable, it is tough to keep track of them. Their messages may just disappear in the flow of communication. Unfortunately, one may not know when to rely on whom.

Meetings: A lot more work might have been done in the number of man-hours used in a typical meeting. Meetings disrupt the "flow" state of efficient software development, forcing engineers to repeat the process of constructing the mental structures required to create or modify the code. Are meetings really necessary? No. Some meetings are worthwhile for attendees. However, how they are constructed is another matter.

If the goal of the meeting is to spread information, it may have been done using an asynchronous medium such as e-mail. Even inquiries concerning the material might be answered by e-mail. If the meeting is about making a choice, the individuals who really need to make the decision are among the vast majority of those there. That is, everyone who attends but does not contribute to the decision-making process is wasting their time. A junction point is rarely so collaborative that a room full of ten individuals has an equal voice in the path that the team ultimately chooses.

I utilize vlogs in this research to learn about the activities that developers value and want people to know about in their professional and personal lives. Vlogs are a richer medium than text-based blogs, which limit the type of content authors may post. Vlogs are a much richer medium than text-based blogs, which restrict the type of content authors may post.

Vlogs allow producers to express their life stories through video and audio that represent their personalities.

Since many different personal narratives are given and the creative autonomy of the creators, developer vlogs can shed light on the comprehensive experience of what it means to be a software developer, which can help dispel prejudices about what software engineers do. There are negative perceptions of what software development "should" be. Working in isolation from others [4], having excellent math abilities [5], and identifying as a nonathletic white guy [6] are all examples. This stigma may discourage the next generation of software developers from pursuing a career that is both collaborative and social [7]. Despite the existence of these negative preconceptions, developers have been working to intentionally remove them through hashtag campaigns [8], written blog posts [9], and podcasts [10]. However, there has been little research on how to vlog creators and viewers might fight these misunderstandings by displaying scenarios in which technical employees have control over their own stories.

To study how engineers present themselves publicly, I collected and analyzed vlog (video blog) data from YouTube, a popular online video-sharing site. I focused on videos portraying a typical day in the life of a software developer. For the research, I identified 30 vlogs with a total length of 700 minutes and over 10 million views.

A popular format for developer vlogs is the "A Day in the Life of a Developer" videos. These vlogs recreate a developer's entire day, from waking up to eating meals, going to work, what they do at work (such as coding, testing, meetings, co-running in teams), breaks, and what they do outside of work (such as running, going to movies, spending time with children and families, or playing games). Each vlog has its own personal tale and emphasizes a number of those activities. This vlog offers the impression that the viewer is physically shadowing the developer without really doing so. Furthermore, the condensed structure of vlogs encourages vloggers to be careful about the content they share, typically optimizing for what they believe will be the most useful to share publicly. Understanding what software developers feel is useful to share as part of their online image, as well as how the public sees software developers, may assist in acquiring a better understanding of what software developers find valuable to share as part of their image.

I observed that vlogs highlighted a wide variety of perspectives at the intersection of developers' professional and personal lives, a whole viewpoint that is difficult to capture through typical data gathering methods like interviews and questionnaires. Vlogs may address subjects such as company culture, work-life balance, and relationships with coworkers by not just talking but also showing. Many pieces of epidemiological research oversimplify software developers' jobs by portraying them as "mathematical geniuses" from a university or "no time for joy" workaholics at a huge technology corporation. Vlogs, on the other hand, reveal that developers take on a range of collaborative roles and approaches to their jobs, as seen by this collection of first-person viewpoints. Freelancing, becoming a lead developer, working at a startup, or switching from a music profession to coding as you go. It is observed that vloggers were able to generate a community of aspiring and experienced software developers who were both encouraged to start and maintain careers in the field. Vlogs allow developers to be open about their careers, changes (such as losing a job), and journeys while also allowing others to discuss similar struggles and offer support.

1.2. Aim of the thesis

The main aim of this thesis is to find out the factors that affect software developers while working in a team. To achieve this goal, I did this two-fold. The first to study the vlogs presented by software developers from different countries about their "a day as a software developer". The reason behind using vlogs, in the beginning, was to identify the various activities done by software developers in their daily work. And the second is to identify those activities or aspects that could affect software developers while working in a team, thus affecting their productivity. Based on the results of the first step, I conducted a survey (personal interviews) to identify factors that can affect software developers while working in a team.

1.3. Research methods

1.3.1. Vlogs

A vlog is the end result of a video-making process and is used to communicate with a large audience. In this research, I study the comprehensive set of software developers' experiences using vlogs. I investigate how developers explain a day in their lives using vlogs. The main aim is to find out and analyze all the activities that can have an impact on software developers while working in a team.

1.3.2. Qualitative research

This involves gathering and evaluating non-numerical data (such as text, video, or audio) in order to better comprehend concepts, views, or experiences. It can be utilized to gain in-depth insights into a topic or to develop new research ideas. By using these vlogs, I did qualitative research specifically to better understand the sorts of materials and how software developers interact with team members, and how they solve any specific problem together (in a team). I opted to investigate software workers who vlog because social media, such as blogs, is a growing resource that software developers utilize to communicate their technical work that overlaps with their personal lives.

1.4. Thesis outline

This section highlights the contents of each chapter in this thesis report. The chapter-wise outline is as follows:

Chapter 1 (Introduction): This chapter introduces the thesis work. Details regarding the background of the thesis, the aim of the thesis, research methods, and thesis structure are provided in this chapter.

Chapter 2 (Related Work): This chapter will provide a review of past relevant research for this thesis work. It shares the detailed work done in the fields of teamwork, team communication, meetings, and VLogs.

Chapter 3 (Data gathering and study design): This chapter provides detailed planning and design of the experiment. It includes a detailed description of gathering software developer vlogs, card sorting survey, and preparation of the survey (personal interview).

Chapter 4 (Analysis 1): This chapter highlights the data analysis of Vlogs and card sorting and interprets the results.

Chapter 5 (Analysis 2): This chapter highlights the data analysis of the survey (personal interviews) and interprets the results.

Chapter 6 (Discussion): This chapter provides an overview of the results and provided a discussion on important aspects of the study.

Chapter 7 (Conclusion and Future Work): This chapter provides the conclusion to this research and the future scope of the research topic.

2. Related Work

This section presents the concepts of team, teamwork, software teams' communication, distributed software development, and distributed agile software development. The concept of teamwork has its roots in the social sciences. One shall discuss teamwork as presented in the social sciences to get a better understanding of teamwork in distributed agile software development.

2.1. Team and Teamwork

Research on teamwork has attracted research from many disciplines [31, 32]. Dickinson et al.[33] have proposed a conceptual model of teamwork consisting of seven components, namely: communication, team orientation, team leadership, monitoring, feedback, backup, and coordination. In their research, measurement items for each of these components are presented through critical incident methodology, where the authors have defined the measurement items for each component by observing the behaviors of the individuals.

Referring to the teamwork models of Dickinson et al. [33] and Salas et al. [34], multiple studies have been conducted in the agile software community to investigate the nature of agile teams. Moe et al. [36] investigated the nature of self-managing agile teams with the help of Dickinson's teamwork model by conducting a case study in a software development company that uses Scrum. Their findings indicate that the specialized skills of team members, as well as the corresponding division of work, were the primary barriers to effective collaboration. Furthermore, they suggested trust and shared mental models should be considered for agile teams apart from the seven components of Dickinson's model.

Knowledge teams, like other teams, must acquire and manage important resources in order to accomplish their tasks. Expertise, or specific skills and knowledge, is the most crucial resource for knowledge teams, yet simply having expertise on a team is inadequate to deliver high-quality work. To maximize the value of expertise, it must be managed and coordinated. To put it another way, teams must be able to successfully manage their skill and knowledge inter dependencies through expertise coordination, which comprises recognizing where expertise is located, understanding where expertise is needed, and bringing needed expertise to be. The study analysis finds that expertise coordination has a high link with team performance that persists even after controlling for team input characteristics, the availability of expertise, and administrative coordination. [16]

An early study by Bannon et al. [15] of the command history of a research group observed that these computer users did not complete tasks in an orderly, linear fashion, but rather took on multiple tasks and switched between them. Czerwinski, Horvitz, and Wilhite [14] found similar behavior in their diary study of task switching behavior, as did Gonzalez and Mark [13]'s study of analysts, developers, and managers at work. Czerwinski, Horvitz, and Wilhite's [14] participants reported an estimated 0.7 interruptions per task and indicated that they not only switched tasks of their own accord but were also frequently externally interrupted. Studies in the workplace have consistently found interaction to be a significant component of knowledge work. Hudson et al. [12] found that managers spent 46 percent of their time communicating with others and 19 percent of their time on unplanned communication. Whittaker, Frohlich, and Daly-Jones [11] tracked two mobile knowledge workers and found that informal interactions were a significant component of the work, comprising 31 percent of total work time. They also observed that these contacts were typically short, context-dependent, and opportunistic. O'Conaill and Frohlich [3], drawing on the same data to look at disruptive, unscheduled interactions, concluded that such interactions were beneficial for the interrupted party but disturbed workflow, as workers often failed to resume the interrupted task. Interruptions were determined to be crucial to work in Perlow's [2] ethnography of software developers, but the large number of interruptions combined with a lack of control over interruption incidence led to employees reporting high levels of stress and frustration.

2.2. Team communication

Interruptions were determined to be crucial to work in Perlow's [2] ethnography of software developers, but the large number of interruptions combined with a lack of control over interruption incidence led to employees reporting high levels of stress and frustration. Aside from measuring and interpreting code and bug counts [45], the availability of publicly accessible communication artifacts has enabled academics to investigate trends in software development in significantly more depth than would be conceivable if just technical artifacts were evaluated [46]. This type of evidence has been used to assist analysis at the individual and team levels, offering insights into participants' contributions to code. Such findings may be found in the literature on communication and coordination from both open-source software (OSS) and closed-source software (CSS) repositories. For example, in the OSS environment, Abreu and Premraj [46] analyzed the Eclipse mailing list and discovered that developers interacted most often during release and integration phases, but that greater contact was also associated with a larger number of defects introduced.

Bird et al. [47] verified that the more software development a person does, the more coordination and managing he needs to do. Cataldo et al. [48] discovered that the practitioners who communicated the most and also contributed were the most engaged throughout software development. Furthermore, Shihab et al. [49] discovered in later research that suggestions and actions mentioned during team communication were connected with subsequent software development activities executed while analyzing the GNOME OSS project.

The generation of insights like the ones just outlined adds to the importance of researching team communications. Indeed, Datta et al. [44]'s SNA research of agile developers' cooperation on the IBM Rational Jazz platform discovered that developers' expressions during frequent contact contained a plethora of important information, far more than could be gained from studying source file modifications alone. Previous research (published in 1994) on the activities of software engineers discovered that up to 50% of practitioners' time was spent on interpersonal communication and collaboration while solving software problems [43].

2.3. Meetings

According to Rogelberg et al. [42], employees spend an average of 5.6 hours per week in planned meetings. While meetings are crucial for good cooperation [40], multiple studies have found that a large part of meeting time is seen as ineffective [39].

Kauffeld and Lehmann-Willenbrock [40] investigated micro-level processes in team meetings to learn more about what makes these meetings effective. Their findings revealed that more functional contact in team meetings, such as problem-solving interaction and action planning, was related to higher team productivity [40]. Furthermore, they discovered a correlation between positive procedural communication in team meetings (i.e., remarks aimed at structuring and arranging the conversation) and team success.

Scholars have researched several meeting elements, such as meeting organization and frequency. Blue Dorn et al. [38] demonstrated that standing meetings may be cut in half without sacrificing the quality of the decisions made. Luong and Rogelberg [37] observed that "the number of meetings an employee had per day was associated with increased daily exhaustion as well as greater subjective stress." Furthermore, Zijlstra et al. [35] discovered that "being interrupted repeatedly had a greater effect than a single longer interruption."

The practice of holding daily meetings in software development teams gained popularity with the advent of agile methodologies such as XP and Scrum, which need daily meetings. Daily meetings are becoming standard practice in agile software projects [32].

The daily meeting is sometimes referred to as the “Daily Scrum” in the Scrum environment [31]. The phrase “stand-up meeting” is also used to underline the idea that all meeting participants should stand in order to avoid lengthy discussions [25]. The meetings should take no more than 15 minutes, with the goal of improving communication, highlighting and encouraging speedy decision-making, and identifying and removing barriers [24]. According to Schwaber [31], team members should address (1) what has been done since the last meeting; (2) what is planned to be accomplished before the next meeting; and (3) what impediments are impeding the completion of task items in the backlog.

In our earlier research, we found that the daily meeting was seen as an important forum for information flow in the project [23] and to defend decisions and show other team members that a great deal of work had been accomplished since the previous meeting [22].

2.4. Software Developer Vlogs

Souti et al. [21] studied and analyzed 130 vlogs presented by software developers and conducted a survey with 335 software developers at a large software company. They found some interesting results. They found that developers were motivated to promote and build a diverse community by sharing different aspects of life that define their identity and by creating awareness about learning and career opportunities in computing. Through the various topics discussed in the vlogs, developers share their experiences from learning programming and their career as a developer in different situations, as well as their journey as a developer in the long run and the effect these career decisions have on the overall quality of life. Also [21], they did comment analysis, and they found that the vlogs were valuable to the audience in finding information and seeking advice.

3. Data gathering and study design

3.1. Data Collection

Qualitative research uses a variety of methods, including interviews, focus groups, and observation [24]. Unstructured interviews may be conducted with open-ended questions on a certain topic, and the interviewer adjusts to the replies. Structured interviews feature a set number of questions that are asked of each participant. It is often done one-on-one and is ideal for delicate or in-depth issues. When group dynamics and collective viewpoints on a topic are needed, focus groups with 8–12 target members are frequently employed. Researchers might be participant-observers who share the subject's experiences, or they might be non-participants or detached observers.

In the first step, I wanted to get information about the daily routine of software developers. In ideal conditions, the best practice is to conduct one-to-one interviews, but due to time constraints, I have analyzed and gathered the information from the vlogs that are published by software developers on YouTube.

After the analysis of the vlogs, the other software developer's insights on the gathered data were required. I used a card sorting survey to find out the activities that have an impact on software developers while working in a team.

Finally, to discuss the factors affecting software developers while working in a team in-depth, I conducted one-to-one interviews with experienced software developers.

3.2. Gathering Software Developer data through Vlogs

In this research, I evaluate how developers describe their day on YouTube using vlogs. According to Statista 2022, YouTube can reach a vast audience of more than 2.6 billion people without charging a fee. These characteristics make YouTube a welcoming and appropriate platform for hosting socio-technical video material. Vlogs, which are generated by developers, are one example of such socio-technical material being generated by developers.

For two main reasons, I decided to focus on YouTube vloggers rather than other platforms. First, "everyone is already on YouTube," with over 2.6 billion people using the network. Second, YouTube has become the most popular platform for vloggers, with recent research using the words "YouTuber" and "vlogger" interchangeably.

I utilized a multistage sampling technique to obtain an initial set of 42 YouTube vlogs that represented countries or regions with a significant developer presence. The initial set of videos were discovered by searching YouTube for phrases like "developer life," "day in the life," and "day in the life + software engineer." The resulting videos included developers from North and South America, Europe, and Asia. I conducted a focused search using the phrases "Germany, Canada, the United States, and India." Other Asian countries, such as Singapore, Bangladesh, and China, were also represented in the findings. I discovered an additional 8 videos via steamrolling (1) YouTube recommendations, which revealed vlogs not included in the original sample, and (2) other vloggers mentioned in our vlog sample. After deleting 12 videos from the initial research (4 videos that were not in English, 5 videos that were recorded more than once, and 3 videos of various categories, such as a recorded conference speech), I came up with a final sample of 30 videos. All 30 videos were uploaded within the last 2 years (January 2020–May 2022) and are from 30 distinct developers. In the rest of the report, these vlogs are referred to as V1-V30.

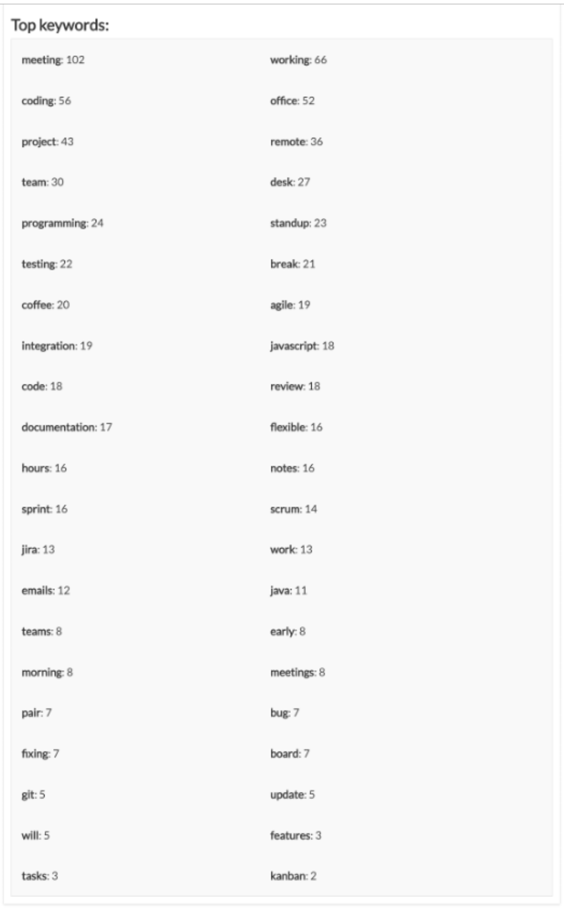
I perceive vlogs as a chance to establish a community of practice in which content creators share a video about their experiences as developers and viewers respond to it. In this context, each video offers a chance for vloggers and viewers to consume, share, and respond to their perception of what it means to be a developer. These vlogs represent a developer's whole day, including waking up, eating, getting to work, what they do at work (such as coding, testing, meetings, and teamwork), breaks, and things they do in their spare time (working out, going to movies, spending time with kids and families, or playing games). Each vlog tells a different tale and stresses different aspects of these activities.

3.3. Card Sorting

I analyzed the transcripts of these 30 videos. I transcribed the videos and emphasized the major factors considered being said by vloggers, as well as the activities they perform as part of their daily lives. I found a few common activities in their vlogs. By using the word frequency calculator, I found that the major topics that are covered by developers in their vlogs can be categorized into 5 major categories, which are, Learning process, Coding / Programming, meetings, working from home/ Remote work, and lifestyle. So, as a next step, I wanted to know about the factors that can impact software developers while working in a team. A software developer's workday may be impacted by a range of variables, including the activities being performed, meetings, interruptions from coworkers, the infrastructure, or the office atmosphere. Some of these characteristics create activity and context shifts, which can lead to fragmented work and have a negative impact on the developer's perceived productivity, task progress, and output quality.

As there is a huge amount of text in the video transcript, it is very difficult to calculate the frequency of any word manually. So, I used the word frequency counter (https://codebeautify.org/word-frequency-counter) to calculate the word frequency.

So, based on this result, I have decided to use the card sorting method to see how the other software developers see these topics. Jennifer Kumar [26] has prepared a list of words and phrases used by software developers in daily conversations at work. I have decided to select the keywords that are in the list of [26] as well as the most frequent words that I got from the video transcripts. I have selected 24 keywords that are related to software development and are mostly used in the vlogs. Ex. The word “meeting” appeared 102 times in the context of daily meetings, problem-solving meetings, etc.



Top keywords:

meeting: 102	working: 66
coding: 56	office: 52
project: 43	remote: 36
team: 30	desk: 27
programming: 24	standup: 23
testing: 22	break: 21
coffee: 20	agile: 19
integration: 19	javascript: 18
code: 18	review: 18
documentation: 17	flexible: 16
hours: 16	notes: 16
sprint: 16	scrum: 14
jira: 13	work: 13
emails: 12	java: 11
teams: 8	early: 8
morning: 8	meetings: 8
pair: 7	bug: 7
fixing: 7	board: 7
git: 5	update: 5
will: 5	features: 3
tasks: 3	kanban: 2

Figure 3.1.: Word Frequency

Card sorting is best regarded as a tool that help to understand the individuals, rather than as a collaborative approach for developing navigation. The method is straightforward. You hand out a set of cards (usually index cards) with sample material printed on them. Alternatively, you might give users a set of content cards as well as a set of categories and ask them to sort the cards into the specified categories. In any case, you document the findings, evaluate them, and apply what you’ve learned to your project. Finally, the third category. During the hybrid card sort, participants are asked to sort cards into categories, but they can use either the categories provided by the researcher or construct their own. It enables them to add some structure to the activity while also allowing the participants some freedom and the ability to express themselves. It also enables the researcher to provide the participant with an example of what they are searching for without totally directing them to (possibly biased) results. As the name implies, hybrid card sorts are more flexible than closed cart sorts. Whether you want to know whether participants prefer your current categories or if new ones make more sense to them, you might want to explore the hybrid option. I chose hybrid card sorting, which combines open and closed card sorting approaches. Participants can sort a set of cards into established categories, but they can also make their own. The hybrid card sorting technique is used when you already have certain categories established but you need input on how the remaining ones should be labeled. The main reason behind it is that it allows the participants a certain level of freedom and the opportunity to showcase their opinions. Also, it is possible to provide an example to the participants with one or more sorting categories. As the name indicates, hybrid card sorts also allow more flexibility than closed-cart sorts. Participants can create their own category to sort the cards.

The next step consisted of identifying experts that would perform as “judges,” who would select the most appropriate items to form the constructs and group them into the predetermined dimensions. Twenty software developers (who are my former colleagues and friends) were invited to participate in the study, but only 11 agreed to participate. All of them were working as software developers with good experience in this field. The reasons for such a choice were based on the assumption that these judges would have good knowledge of what is involved in academic research but also have sufficient acquaintance with the details and peculiarities underlying the industry.

The judges were contacted personally and informed of the purpose of the study. Then they were asked to choose those items they judged more appropriate as an indicator of the construct and to group each item into any of the predetermined dimensions, putting them into the corresponding column.

3.4. Survey

The success of any software development project lies in careful planning, a skilled development team, and clear communication among team members, both internally within the software development company and externally with the customer or product owner. And a software developer is the core component of a team. Identifying the factors that influence the performance of team members is crucial in the software development industry because software development requires a team effort. Thus, the purpose of this study was to examine the factors that influence the performance of software developers while working within a software development team.

The survey focused on software developers' team activities. Based on the vlogs' analysis and card sorting results, I have designed a questionnaire for conducting interviews. As per the vlog analysis, I found that the activities that can impact a software developer's efficiency include coding and various meetings. I found that from the first study results, the categories "Meeting, Review, and Team" contain those activities. I decided to use them as a part of our interviews.

Several questions were concentrated on these three categories of activities that developers conduct daily. The list of activities was compiled from the 5 categories described in the vlogs. I included five distinct activities from the five categories. The survey was divided into three sections. The first section consisted of questions on personal information. The second section inquired how much time developers spend attending meetings. Finally, the final section inquired about developers' coding routines and pair programming.

Initially, I planned to prepare a survey and share the link on social media platforms such as Twitter, but due to a time barrier, I decided to focus on the quality of the result instead of quantity. Interviews are a good way to both cover a range of topics and make sure that important information is not missed in a one-to-one interview. They are an effective method for providing reliable, comparable qualitative data with different participants, even if they are given different interviewers. The level of conversation that interviews allow can provide new perspectives and comprehension of the subject at issue.

Interview design:
Generally, interviews are conducted either way, with an individual or with a group of people, called focus groups. In this research, I have conducted individual interviews where interviews are done one person at a time. Initially, I planned to voice record the interview session, but due to privacy concerns, I decided to write down the notes during the interview and later elaborate on them. Interviews were conducted in English, transcribed, and finally summarized and structured for further analysis. The following are

the features of the interview that I conducted:

Open-ended questions: Through these questions, I aimed for an extended discussion of the topic instead of just a yes or no. In this way, interviewees had the freedom to express their opinions based on their experiences.

Semi-Structured format: I focused on getting an in-depth knowledge of the topic through interviews. This can be achieved if the interviewer has a set of questions and issues that are to be covered in the interview and asks additional questions whenever required. Due to the flexible format, I have chosen semi-structured interviews.

3.4.1. Interview Questions

1. What is your job profile?
2. How long have you been working as a software developer?
3. How do you feel about working in a team environment?
4. Have you ever had difficulty working with a manager or other team members?
5. Why is proper communication important in a team?
6. What are some problems in a software development team's communication? Where, according to you, can communication go wrong?
7. How many hours do you spend attending meetings?
8. Which of the following meetings do you participate in? Brainstorming meetings, Check-in meetings, Decision-making meetings, One-on-

one meetings, Problem-solving meetings, Quarterly planning meetings, Team-building meetings?

9. How do these meetings impact your quality of work as a software developer?
10. Are there any other thoughts related to team meetings?
11. How many hours do you spend working with code or programming (e.g., implementing, bug fixing, adding features) in a week?
12. How does the interruption from another coworker affect your work? How frequently does it happen?
13. What are your thoughts on pair programming?
14. What are the advantages and disadvantages of pair programming?

4. Analysis 1

4.1. Analysis of Vlogs

In our study of vlogs, I discovered that developers described experiences at the interface of their professional and personal lives. I transcribed the videos and emphasized the major factors considered being said by vloggers, as well as the activities they perform as part of their daily lives. I found a few common activities in their vlogs. I outlined five categories: their learning process; programming; meetings; home office/remote work; and lastly, their lifestyle. But I am going to focus more on meeting and coding/ programming categories as activities performed in these categories require team efforts. Although teamwork has been widely researched in a variety of domains, little of this information has been utilized in the context of software development and negligible information. One of the reasons might be that broad knowledge must be customized to software development in order to be beneficial. As a result, there is a need for further research on teamwork in this specific field. Also, I noted some additional findings at the end of the analysis. They are not covered in this research but can be useful for future work.

1. **Meetings**

   Meetings were a significant work-related activity. Of the 30 videos, all 30 developers (video creators) talk about their daily meetings. Marsela [V1] explains that she has various meetings during her workday. Most of them talk about their daily meetings or stand-up meetings (report meetings) in which they explain their progress on the assigned project or task. There were three kinds of meetings: status meetings, kick-off meetings, and spontaneous meetings. Status report meetings, which include daily scrum-style stand-ups or individual check-ins with a manager, were the most prevalent of the three [V27]. These sessions gave developers the opportunity to seek assistance. "Having a stand-up in the morning absolutely helps with that because it gives you an opportunity to say, 'hey, I'm having a small problem with this part'". "Can I get any assistance? and others are eager to help." [V6]. Project kick-off meetings allow new project team members to "understand precisely who else is on the team... to not plan work but how we want to work; what type of meetings we want; how we want to monitor our work, and that sort of thing" [V19]. Later in the project life cycle, videos showed developers together planning project sprints at the start of a project or conducting retrospective sessions at the end of a project. Unscheduled meetings spurred by a colleague coming by their desk to clarify a task or a spontaneous cluster of coworkers around a developer's code on the screen were examples of other forms of meetings. This wide range of sessions demonstrated the breadth of communication that occurred within a team. Here are 3 common barriers to effective communication.

Defensiveness:- You take any form of criticism/ advice as an accusation that you must prove wrong at all costs. You take it as a form of disrespect and a hurtful statement that you cannot accept. Hence, it's impossible for you to actually take the advice into consideration and make a change.

The wrong objective from discussions:- Continuing from the last point, taking discussions as battles where you either defeat or get defeated. You want to win this argument and prove the other party wrong instead of looking at it as an opportunity to learn and a chance to see things from another perspective and reach some sort of middle ground.

Bad listening: Listening isn't the same thing as hearing. Listening means paying attention to what the other party is saying, grasping it, genuinely considering it, and trying to understand it.

2. **Coding and programming**

   As expected, developers are present in the vlogs doing code and other coding activities as part of their profession. There are different ways to demonstrate coding, but the most common is a time-lapse video of the coding session with the displays blurred. Once developers have determined which job they will work on, they must become acquainted with the appropriate artifacts and code. They study the ideas in advance, such as "authentication"[V12], or while they work on the tasks, which include implementing, testing [V4, V9, V22], debugging [V24], executing and building, and releasing and deploying code to deliver new features or patch defects [V13, V28]. They also demonstrate activities such as reading task-related concepts [V21], planning and drawing out a solution [V29], looking for answers on Stack Overflow or GitHub [V14, V18], and interacting with repositories [V14]. When not actively implementing, developers evaluate code, which is defined as "essentially a commit that somebody submitted, and they are forwarded for review in our team just to figure out if there are any flaws and issues that happened in the code." [V8]. The videos typically include code reviews. Code reviews are sometimes high-priority activities [V27]. The videos also showed the team examining pull requests from other teams [V27], interns [V18], and younger team members.

3. **Learning process**

Vlogs also provided advice and suggestions on how to get started in the field of programming by deciding which programming language to use. This happens all the time. It begins with which programming languages are excellent initial languages to learn [videos 1, 5, 6]. Vlogs also explained that the programming language you select is frequently determined by the task at hand and provided instances from their professional expertise. For example, one vlog describes how they decide which programming language to use: "In terms of what coding languages I use at work, it really depends on the assignment." My team operates across many business divisions, so we're always switching. I mostly utilize JavaScript for our node-based apps, but I also use a lot of Java [V7]. According to another video, programming languages such as C# "assist in grasping essential concepts of a programming language such as data types, object-oriented control statements, data structures and algorithms, reading network protocols, and the graphic debarment." Vlogs go on to detail the many responsibilities within the development, such as the front-end, which requires "HTML, CSS, and some JavaScript" [V21], and the back-end, which requires "C#, SQL, and other databases like MongoDB" [V22] [V21]. This vlog also discusses the advantages of being a full-stack developer, capable of working on both the front and back ends, as "having that jack-of-all-trades is excellent because you can work at a smaller firm where you can be very effective and add a lot of value to the table" [V21].

4. **Work from home/remote work**

   In their vlogs, developers discussed working remotely and the benefits that come with it. The developers were able to work from several places thanks to remote work. Some developers prefer to work from home at "their own" workstations and equipment [V23]. This reduces time spent driving to and from work [V28]. Others, on the other hand, may "just take the laptop and pretty much go to other places and simply work from there" [V16]. Working from Florida rather than Chicago in the winter [V26] or working from Berlin for a change of scenery [V30]. When discussing working remotely, developers frequently mentioned how "easy it is to get distracted" when working from home [V21], and thus many frequently consider working spaces [V20] or finding coffee shops [V24] or public lounge spaces [V28] to work from in order to stay productive.

   COVID-19 and Quarantine Because I did our video analysis in the summer of 2020, our sample of vlogs includes videos in which developers discuss how they work during a worldwide pandemic and the added productivity issues they experience during a crisis. Many developers who were already working remotely and some who were moving to remote work expressed their problems in these vlogs. Non-ergonomic work environments [V18] and obstacles to their mental well-being [V26] were among these problems. Specifically for free-lancers, the pandemic resulted in project cancellations since some firms didn't have the money to pay all the engineers and remote engineers [V23]. The videos also highlighted suggestions on how teams were overcoming problems, such as staying socially engaged with their colleagues through "remote lunches" [V30] via video chat. In some respects, these vlogs serve as a historical capsule of how software engineers functioned during that difficult epoch of remote work.

5. **Lifestyle**

Healthy living and well-being Some vlogs stressed the significance of maintaining a healthy lifestyle, which includes frequent workouts, and led by example by displaying various activities they participate in via the vlogs. In these video parts, developers discussed the importance of developing long-term habits such as staying hydrated [V26] and practicing meditation [V18]. Many vlogs also include physical and social activities such as discussing exercises, nutritious recipes, and eating habits to illustrate how they live a healthy lifestyle. Developers stress the need to take frequent breaks during the day to go for a walk, go to the gym [V11], [V14], [V18], or participate in team sports with friends, such as tennis [V7], basketball [V13], and martial arts [V19].

4.2. Card sorting Analysis

I created a link and sent it to my software developer friends, and asked them to sort the given cards, which contain activities performed by software developers. And I got 11 responses from them. As a result, I received 11 different categories, of which 2 were defined by us (Programming and Team) and the other 9 were defined by the participants, which were: comfortable with/in, meeting, project management, review, skills, time management, team leader, the first thing to do, meet in person.

I am focusing on only three categories that are related to team management: meetings, review, and team. Here are the card sorting results for these 3 categories.

According to the meeting category result, all 11 participants attend daily standup meetings. Also, 8 participants are working in an Agile-Scrum methodology, which has different types of meetings like sprint planning, retrospective, sprint review, and backlog refinement, which consume software developers' time. Also, code review (3), pair programming (5), and knowledge transfer (6) are important team activities that are sorted in this category.

According to the team category, 4 participants think "bug fixing" should be done in a team. On average, six participants sorted the activities that are performed after finishing with coding into this category, e.g., bug fixing, code integration, code review, pair programming, and testing.

Based on these results, I choose the whole meeting category, pair programming, and knowledge transfer activities, which can impact more on the productivity of software developers while working in a team.

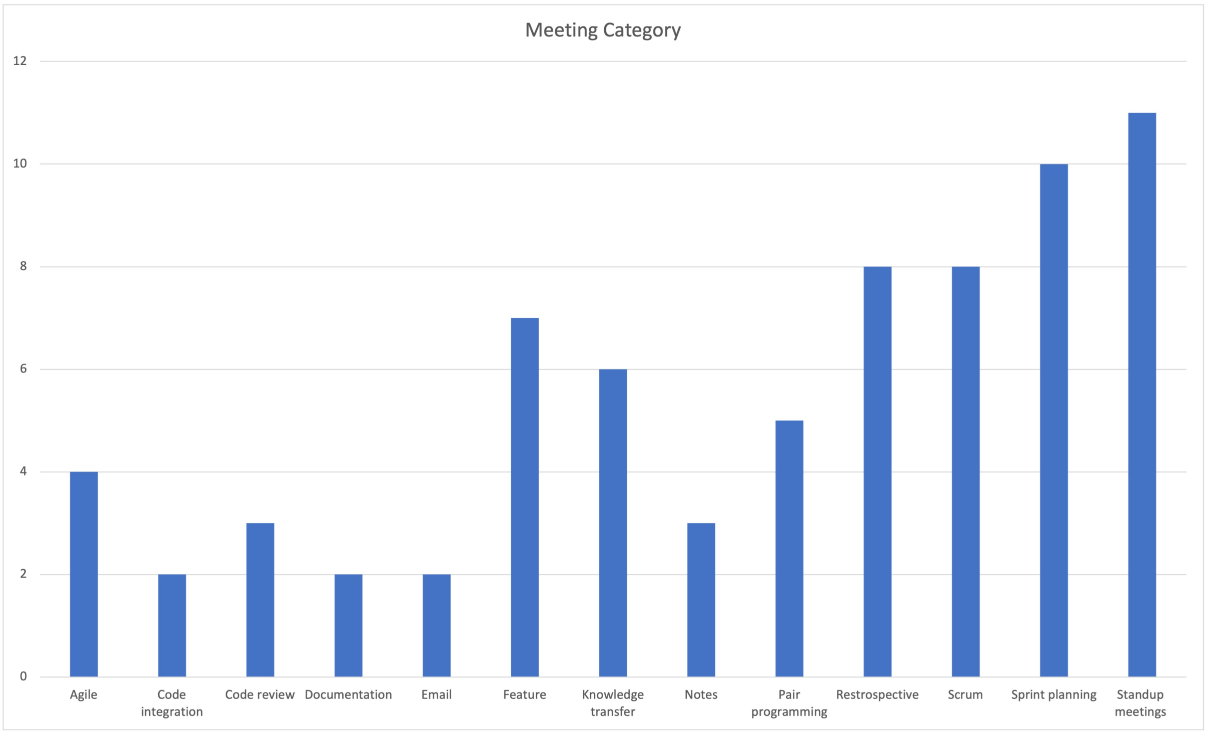


Figure 4.1.: Card Sorting Result - Meeting Category

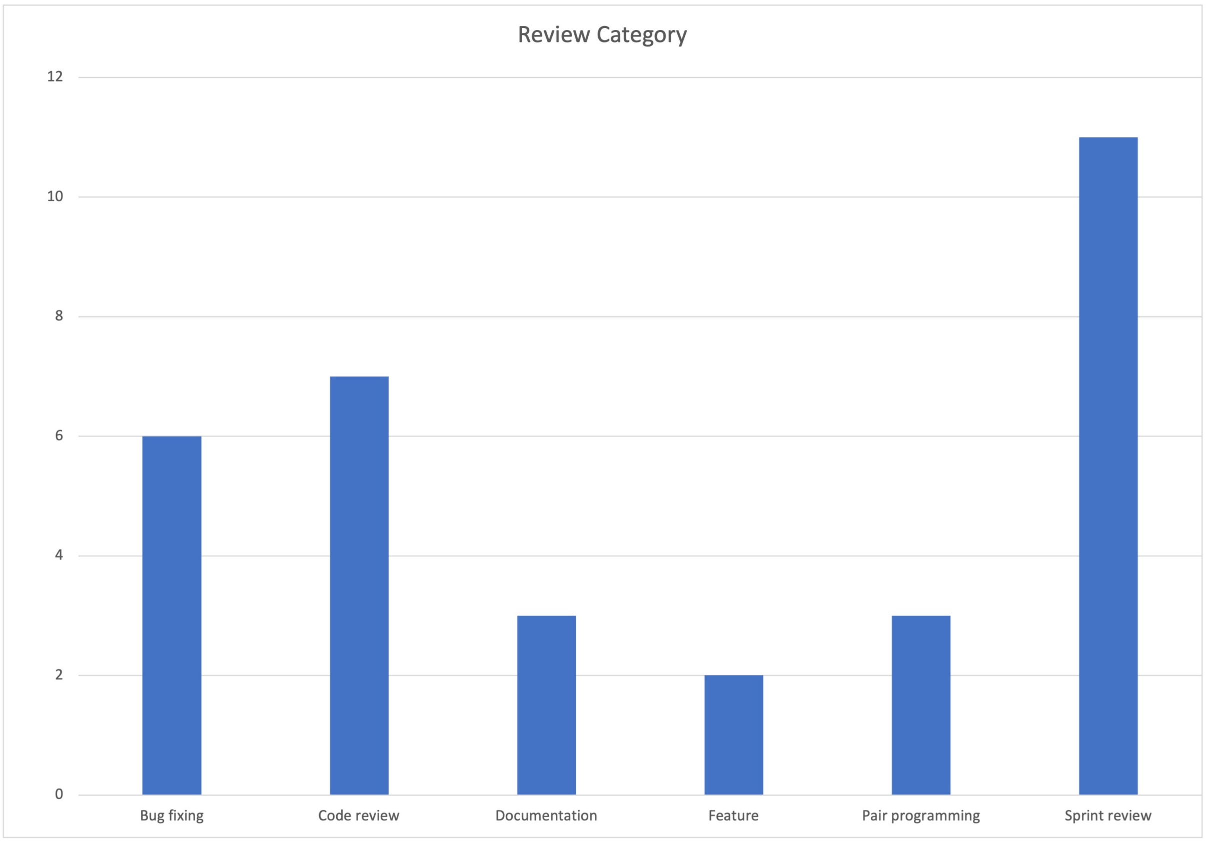


Figure 4.2.: Card Sorting Result - Review Category

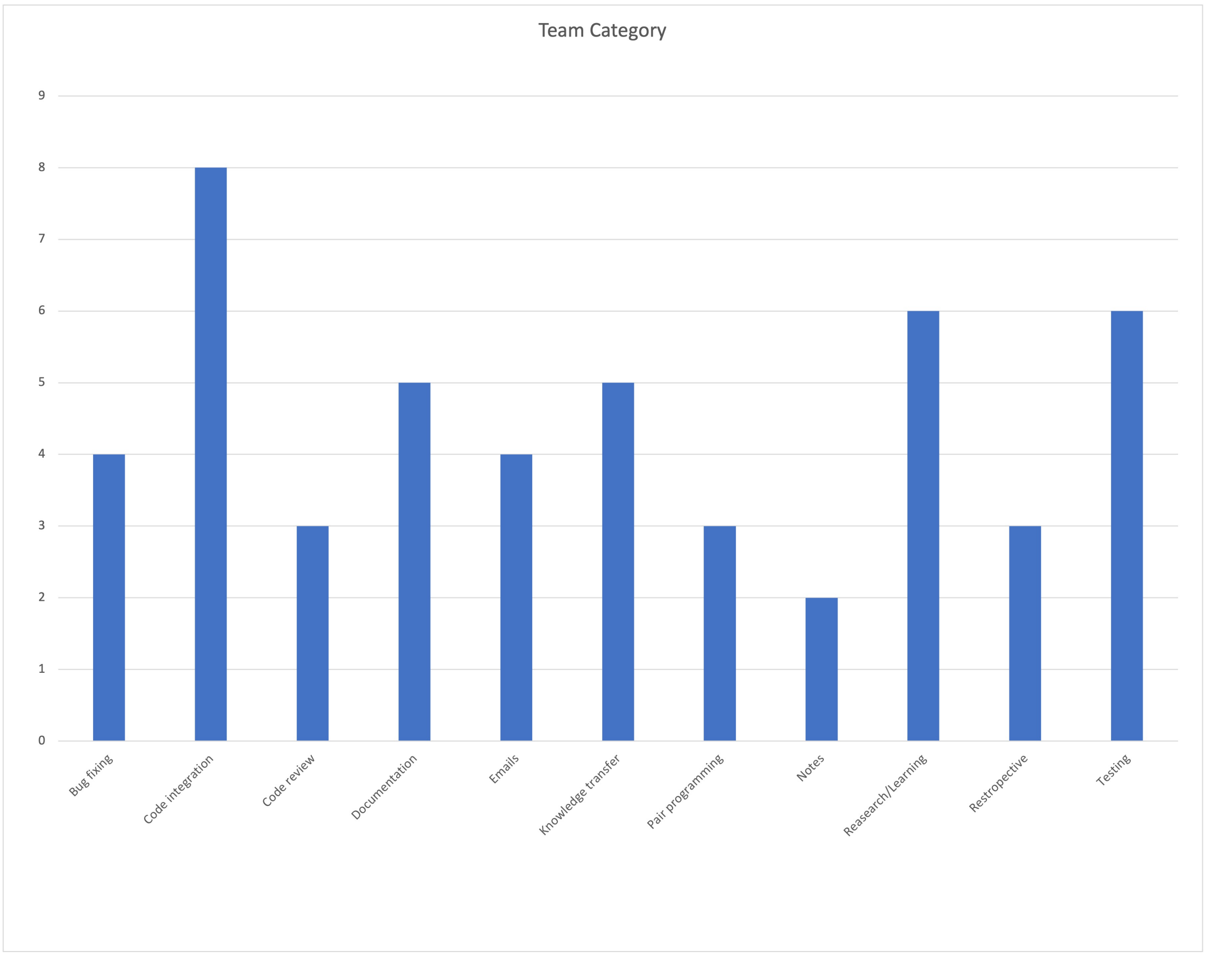


Figure 4.3.: Card Sorting Result - Team Category

5. Analysis 2

5.1. Survey interviews Result

Here is list of Software developers who participated in the one-to-one interviews. To identify them I have named them as P1, P2, P3, P4 and P5.

No.	Gender	Job Title	Age	Experience (in years)	Working Country	Company Location
P1	M	Software Developer	47	3	Germany	Germany
P2	F	Full stack Developer	29	7	Germany	Germany
P3	M	Backend Developer	27	5	India	India
P4	M	Full stack Developer	32	10	Germany	Germany
P5	F	Frontend Developer	28	5	India	India

Table 5.1.: Interviewee List

5.1.1. Working in team

To envision, design, develop, and implement software that supports corporate processes and functions, software development teams are made up [28]. These teams work in rapidly changing, knowledge-intensive situations where knowledge exchange and integration are essential for success. Furthermore, the essential knowledge, skills, and experience for software development projects are too vast and complicated for any single developer to possess. As a result, software development projects need the participation of various team members with a wide range of knowledge, skills, and expertise [29]. Cooperation between team members gets increased when the quality and acceptance of ideas are high within the team. Therefore, mutual support can be considered an important element of teamwork that enables the team members to achieve the team goals in an efficient and effective way.

The cross-functional component of software development teams refers to the distribution of project expertise among several cross-functional members. Team members may face task conflict during the project if there is no shared understanding and common information. [P1] [P2] [P4] [P5] In this sense, task conflict refers to conflicts and opposing viewpoints and attitudes about a certain task.

Simply being able to discuss a problem with someone who is also familiar with the domain and context helps a lot. Enables pair- and mob-programming when useful. Software development is a very diverse field. Being proficient in all things a project might involve makes it hard to become good at anything. Or you will likely deliver sub-par products in some ways. Having other developers available to review your code helps identify and fix bugs before launching new versions. [P1] [P2] [P3] [P5] Having to coordinate development through version control (and possibly CI/CD) also increases the likelihood of proper version management, making rollbacks and troubleshooting easier.

Working alone can be a fun challenge though, but I find it best suited for personal projects where you can experiment more for fun and just try new technologies with your own ideas. [P4] For professional work, all projects should have at least 2 developers to enable discussions and code review.

"Every teammate has their own education, experience, and skills. It is only natural for expertise to be exchanged when working together. When developers pair up, knowledge sharing is encouraged through code reviews and walkthroughs, retrospectives, mentorship, and continuing education programs". [P1]

"Not only is information exchanged, but coding quality improves when a coding standard is maintained and enforced. The idea is to establish 'good enough code' and utilize code reviews to regularly enforce the adoption of style standards. Even from scratch, it should only take a few weeks for everyone on the team to start using the standard rigorously. Code will be simpler to understand, defects will be easier to identify during the life of the project, which can span a decade or more".[P2]

"Working together towards the common goal of completing a functioning, finished product can increase efficiency. When teammates encourage and help each other to finish a product, it can reduce production time and prevent mistakes that might occur during project completion".[P3]

"The most important thing in a team is to review each other's work. Re-

viewing other developers' code means reading other developers' code can help in improving my skills".[P4]

"The belief that we can do almost anything because we believe in each other. It's amazing how we can comprehend one other with just a few seemingly unimportant words. Furthermore, we help each other improve. And we demonstrate our appreciation and regard for one another in subtle ways". [P5]

5.1.2. Communication

Communication is a critical component of cooperation because it allows team members to share information, coordinate activities, and provide feedback. The correct information should always be sent to the correct person via an effective communication channel, without changing the original message [P3] [P4] [P5]. If the message is distorted while being transferred, it may fail because erroneous information was delivered. As a result, a feedback system is beneficial so that both the sender and the recipient are aware of the validity of the communication. When team communication is of high quality, it serves as the basis for other factors that drive team performance [P2]. As a result, communication may be seen as an essential component in the formation of a high-performing team.

Developing a product entails several procedures, phases, and iterations. Other team members must understand each stage in order to contribute. For example, they must comprehend what you have accomplished, how it will operate, how it should be implemented, and what problem it will address.

Communication is a fundamental talent that all people who want to be successful in their jobs must possess. Though there are many talents that software engineers must learn, communication skills are essential for survival and advancement. [P1] [P2]

When it comes to team cooperation, a strong communication skill set is vital for project success. Poor communication frequently leads to failure and increases costs. Using the developer's technical expertise, the team works on a few challenges to make the end user's life simpler. All products/services must assist them in a way that adds value to their lives. As a result, excellent communication is essential both inside the team and with consumers. [P1] [P2] [P3] [P4] [P5]

Organizations fall into a trap as technology advances, particularly in the software development business, where we just enjoy the new great collaborative tools that come out. Communication is scattered among an excessive number of communication channels, including emails, Skype, Slack, Flock, Microsoft Teams, sticky notes, interesting new collaborative tools, and so on. While the communication aspect remains valid, we must recognize that information disseminated over too many channels is difficult to track. Furthermore, you cannot expect all of your team members to keep track of all those channels and remain on top of every piece of information while focusing on their core work. [P2] [P4] [P5]

Another consideration in communication is availability. When their coworkers do not work from home, individual employees do better. The amount to which coworkers work from home appears to have an impact on the individual employee's performance. The greater the percentage of coworkers who work from home, the lower the employee's performance. Working from home causes employees to become more focused [19]. Because teammates are not always available, it will require greater effort on the part of the individual employee to put their talents and expertise to use. [P1] [P3] [P5]. Also, at some point in time, some component of a task is dependent on a coworker's task, but he or she is not yet through with the task. In that situation, you must wait until he or she has finished it. This dependence issue arises several times when working in a team on the same project feature. [P1] [P3] [P4] Although working from home does not imply that employees are absent, these findings support the notion that digital presence cannot really compensate for bodily presence when evaluated from the perspective of coworkers. This is also determined by the type of job.

"Without communication, there's no foundation for teamwork. And without teamwork, there's no team. You essentially just have individuals in the same department working separately towards a common goal". [P1]

"It contributes to the development of a harmonious code of conduct in which general morale and the substance of our moral principles are remarkably experiencing an ever rising unique sense of belonging to an ever bigger collective uniqueness of shared insights for the greater good. Furthermore, it teaches us to observe ever more exceptional rules of conduct in all that we are ever focused on and putting out in an ever more remarkable manner". [P2]

"Open communication and collaboration are conducive to increased learning opportunities and creativity.When teammates feel comfortable expressing themselves, it leads to the introduction of new ideas, concepts, and processes. This can create changes in the final project and innovation where ideas may not have arisen when working independently". [P3]

"Teams that communicate execute tasks faster and more efficiently than others. They are also more precise than others in their profession. Effective communication also helps team members to understand their own duties as well as the roles of others in the team". [P4]

"Without communication in a team, it is impossible to grasp how far each member has come and how much potential everyone has." In the end, it will result in a disordered team that is less capable of achieving goals. Effective communication also helps team members to understand their own duties as well as the roles of others in the team". [P5]

5.1.3. Meeting

Software engineers meet in a variety of ways and with varying meeting styles. A software engineer meeting is held to share information, collaborate on projects or problems, debate new initiatives or assess progress on previous ones, and achieve a consensus among colleagues. Meetings are used by software engineers in three ways: team meetings, walk-thorough, and face-to-face meetings.

All the developers from the vlogs are attending at least one meeting daily, but according to V7, V19, and V23, they spend more time attending meetings during their day, so I decided to ask our participant interviewees about how it affects their daily schedule of work when there are multiple meetings in a day. So, I got some feedback from them as follows.

For business purposes, such as knowledge transfers and bug triage, we require standard meetings. Basically, any meeting that is necessary to complete your task is acceptable. Certain meetings take place because people do not collaborate as they should, which disrupts our momentum. [P1] [P4] [P5] All of the participants stated that they feel better about their workday when they can spend parts of it working on code rather than meetings, distractions, or administrative responsibilities. While meetings were frequently viewed as a waste of time, [P2] [P3] [P5] answers defined a productive workday as one in which they engaged in excellent, constructive talks, made key project choices, or developed contacts that would be useful in the future. Because developers frequently regard meetings and interruptions as unproductive, previous research indicated that they are poor in general. [27] These earlier findings are being refined by our findings, which show that the impact of meetings and interruptions on productivity varies depending on the project phase. Participants regarded them as "common yet beneficial" during the specification, planning, and release phases. Not at the development stage, but in the planning stage. *"I spent way more time in meetings than normal, but they were productive meetings".* [P4]

Meetings appear to be harmless, and many managers believe they have done something by putting their staff on the same page and informing everyone on the status of current office initiatives. Requiring engineers to attend meetings throughout the day, on the other hand, is counterproductive if the aim is to generate exceptional software. Meetings are detrimental to developer productivity and morale [P2] [P4] [P5]. They are a waste of a good developer's time. The fewer meetings that developers are forced to attend, the better the software generated and the morale of the development team.

According to research done by [20], the average wasted time per significant interruption is 23 minutes. It's considerably worse for developers since you can't quickly return to where you were before the interruption. You must first adopt a development attitude before gradually returning to where you left off. This may take more than 30 minutes.

Meetings usually have one or two purposes: to convey information or to explore options such as potential solutions to problems or future efforts for the company. There are certain meetings that happen because people do not collaborate as they should, and it disturbs our momentum. Meetings can be divided into 7 different categories, such as

- Brainstorming meetings (e.g. product development, Sprint) [P1- P5]
- Check-in meetings (e.g., project status update meetings, Daily/Weekly team meetings) [P1- P5]
- Decision-making meetings (e.g., final approval of a design) [P3] [P5]
- One-on-one meetings (e.g. Client sales meetings, Coaching, or Mentoring sessions) [P1] [P2] [P5]
- Problem-solving meetings (e.g., testing code, bug fixing, code review) [P1] [P4]
- Quarterly planning meetings (e.g., project planning, strategic planning) [P2 - P5]
- Team-building meetings (e.g., virtual team challenges, internal TED Talks) [P4]

Interruptions from co-workers

When someone wants to discuss something with you, you don't have to drop your current work immediately and listen to them [P3] [P4] [P5].

Interruptions from co-workers and distractions such as background noise in open-plan offices were found to negatively influence developers' ability to focus or work as planned. *"Too many interruptions/context switches I need a continuous block of time to be really productive as a coder, but I find I get distracted/interrupted more than I'd like".* [P3]

"Meetings interrupt the flow state of effective software development, forcing me to restart the process of building the mental constructs needed to create or modify software". [P1]

"I can see the benefits, in as much as two heads can be better than one sometimes. Sometimes you can stare at a page of code for a day and be none the wiser as to how to proceed - an extra pair of eyes on the code can be useful in that sort of situation". [P2]

"It is a lifeline and a killer at the same time. Standard meetings like knowledge transfers and defect triage make sense because we need them for work. Basically, any meeting that is required to do your work is fine. There are certain meetings which happen because people do not collaborate as they should, and it disturbs our momentum". [P3]

"I could see using pair programming for particularly hairy, complex design tasks. Any time you're working on architecture, having two brains to bounce ideas back and forth is great. You're less likely to miss important details that way, and for design work, an important detail can mean a lot of code refactoring or rewriting if you don't catch it until late in the game". [P4]

"All meetings have a cost. They interrupt work in progress. They may require preparation. They take time to attend. They may distract workers

after the fact. They may require time to process the information transmitted. Different people will have different opinions on how much these costs are and how valuable the result is". [P5]

6. Discussion

This section summarizes the answers to the research questions, presents implications for research and practice, and discusses the limitations of the thesis.

From my three studies, i.e., Vlog analysis, Card Sorting Survey, and One-to-one interview of software developers, I found that software development by itself is a complex and daunting task. The quality of software also depends on good teamwork, specifically with respect to the interaction processes within a team. Since software development is primarily a team effort and the software developer is the key person who actually implements the functionality, it is important to understand the factors that influence software developers while working in a team.

Communication plays a vital role in ensuring that the various development activities work in concert toward producing a software product in an efficient manner. Good communication in software engineering saves time, reduces errors, maximizes the available budget, and creates a better work environment for teams to be successful. According to the study findings, it was evident that frequent and effective communication among team members positively influenced team performance. A software development project's success is dependent on effective communication. In projects, both formal communication (i.e., scheduled meetings and written status reports) and informal communication (i.e., quick phone calls and short emails) are important whenever they are required. According to [1], a lack of communication creates recurring problems related to the development effort, and to improve communication among members of a software development team, an effective process and the infrastructure to support it must be provided.

The company can have multiple branches, and the team can be spread across the world. Also, nowadays, companies are allowing their employees to work with flexible timings. This is causing an availability/dependency issue. Expertise has been recognized as an important factor influencing the success of software projects. It is also acknowledged in agile software development. Expertise dependencies were found in all of the case projects. An expertise dependency is characterized as a circumstance in which technical or task knowledge is known exclusively to a single person or group, and its absence has an impact on project development. From the interview results, I found that when junior developers have some issues during their tasks, they face the problem of communicating with their seniors as they are not available at that time due to "flexible working" timings. which results in the wasting of some working hours of the junior developer.

Effective communication is crucial in software development. Meetings that are well-planned and handled may be quite beneficial. There are several ways of determining whether or not a meeting is required, and each team is different. From the Vlog analysis, Card Sorting Survey, and One-to-one interview results, I found that all the participants work in an Agile environment, either Scrum or Kanban. The literature usually describes daily meetings as a place for answering the three Scrum questions. However, I found that, while the daily meeting was said to focus on answering the three Scrum questions, most of the meeting was spent on other types of discussions and resulted in spending more time than the defined 15 minutes. There's nothing like a bad meeting to ruin productivity, and sometimes the day. Meetings frustrate software engineers since they interrupt their flow, and hence they see meetings as major distractions that reduce productivity. Meetings are also viewed as unproductive, time-consuming, and counterproductive to their work schedule.

In Pair Programming, two people jointly solve one single task. One person is leading the work as the driver, and the other is constantly overlooking what is happening as an observer. The observer is constantly watching and supervising the present task while he also tries to find a solution for the next task. A pair switches between being driver and observer, and they also change people between other pairs. Pair programming can be used in every development task, but it is mainly used in coding. Half the time, doing one task by two people is not redundant because two people usually find a solution quicker. They also implement it faster and produce it with higher quality. But from the survey, I can conclude that there is too much routine coding and easy tasks, so pairs should produce a maximum of 50% of a project's code. In general, respondents think pair programming is only good when something becomes complicated; otherwise, they think it is too time-consuming. If a special section or task is complicated, they can see a quality increase effect by using pairs, but otherwise, people state that they work faster as singles. Respondents agree that pair programming is a good technique to reduce defects and increase quality. But it is too time-consuming and it cannot reduce other tasks enough to reduce the total development time.

Cooperation between team members gets increased when the quality and acceptance of ideas are high within the team. Therefore, mutual support can be considered an important element of teamwork that enables the team members to achieve the team goals in an efficient and effective way.

Based on the result, 3 hypotheses are generated,

1. More than two meetings in a day can have an impact on a software developer's efficiency in daily work.
2. Any type of interruption (from co-workers, or team leaders), may affect the productivity of developers.
3. Pair programming improves the quality of the code.

Limitation

There are some limitations to this study. First, this study was limited to 30 Vlogs, 11 card sorting participants, and 5 One-to-One interviews with software developers. Due to time constraints, it was challenging to get in touch with more software developers.

For card sorting, some participants were experts who had more than 5 years of experience, and some participants were relatively new to the industry or were students. Experts may have a different impression of a specific card (activity) than the student or new person, who might not have any knowledge or have just theoretical knowledge. This can give us a biased result.

As I performed the card sorting survey online, there was no way to communicate with participants other than by writing a short introduction and instructions before the start of the survey. It is possible that for some participants, the provided instructions are not sufficient enough to perform the survey, and hence that can have an impact on the quality of the result.

Also, due to the online card sorting survey, there is no way to get information on why participants sort the cards the way they do because it is not possible to see the participants or hear them thinking aloud.

The above-mentioned results are the outcomes of the interviews of five participants and the analysis of 30 vlogs. Adding on the number of participants will shed light on different opinions or aspects regarding various aspects affecting software developers and might result in a broader scope.

7. Conclusion

The purpose of this thesis was to investigate the factors affecting software developers while working in a team. The research consists of three phases. First, qualitative analysis of vlogs, card sorting, and one-to-one interviews of professional software developers.

The findings suggest that teamwork is often considered one of the most important "generic skills" that we can provide to graduates entering the information technology profession. From the analysis of the vlogs and interview data, we found that software developers interact with their team at least once during the working day as they use the agile software development method.

We discovered many issues that software developers experience while working in teams, including communication within the team, availability/dependency, interruption, and meetings. When developers work in a team, it makes it easier to review coworkers' work and give or take feedback from them. Communication with colleagues is a more essential source of knowledge than written documentation since written documentation is non-existent and some developers prefer direct communication over writing documentation. We found that developers use multiple communication channels while working in a team. As information spreads across too many channels, it becomes hard to keep track, leading developers to miscommunication. Also, working from home leads to the intensification of employees. Because coworkers are not immediately available, it will take more effort on the part of the individual employee to make use of their skills and knowledge. We also noticed that frequent communication among team members has a positive effect on software development team performance, so the communication can be either personal or through any communication medium.

The next factor we noticed was meetings. According to the result, software developers have at least one meeting a day. Most programmers and developers view meetings as a waste of time. The truth is that a lack of meetings could result in professionals pulling in opposite directions, resulting in a massive waste of time and resources. However, there are several steps that can be taken to make meetings more aligned with the programmer's schedule and enhance their efficiency. Long meetings can interrupt the working flow. Another factor that affects software developers is interruptions when team members ask for help or when bottlenecks occur. While software development teams benefit from knowledge sharing and a shared mindset within the team, the amount of direct communication is high and causes interruptions.

Regarding pair programming, In general, participants think pair programming is only good when something becomes complicated; otherwise, they think it is too time-consuming. If a special section or task is complicated, they can see a quality increase effect by using pairs, but otherwise, people state that they work faster as singles. We also found that, agree that pair programming is a good technique to reduce defects and increase quality.

Bibliography

[1] B. Curtis, H. Krasner and N. Iscoe. *"A field study of the software design process for large systems"*. Communications of the ACM, vol.31, pp.1268-1287, 1988.

[2] Perlow, L. *"The time famine: Toward a sociology of work Time"*. Administrative Science Quarterly, 44 , 1 (1999), 57-81.

[3] O'Conaill, B. and D. Frohlich. *"Timespace in the workplace: Dealing with interruptions. In Proceedings of the ACM conference on human factors in computing systems (CHI'95) (Denver, CO, May 7, 1995)"*. ACM Press, 1995, 262-263.

[4] Garrick Saito. *"What are stereotypes about software engineers that are simply untrue"*. Retrieved October 15, 2020 from https://www.quora.com/What-are-stereotypes-about-software-engineers-that-aresimply-untrue-for-the-most-part.

[5] Stephen Sinco. *"How to Become a Software Developer: The Top 6 Myths Holding You Back"*. Retrieved October 15, 2020 from https://www.codingdojo.com/blog/5-myths-about-how-to-become-a-software-developer.

[6] Armando Pantoja. *"Stereotypes in IT: Armando, you don't LOOK like a software engineer!"*. Retrieved October 15, 2020 from linkedin.com/pulse/armando-you-dont-look-like-software-engineer-armando-pantoja.

[7] Benton Rotchester. *"Can we please stop stereotyping developers"*. Retrieved October 15, 2020 from https: //medium.com/re-write/can-we-please-stop-stereotyping-developers-63a870ea2857.

[8] Fannie Liu, Denae Ford, Chris Parnin, and Laura Dabbish. *"Selfies as Social Movements: Influences on Participation and Perceived Impact on Stereotypes"*. Proc. ACM Hum.-Comput. Interact. 1, CSCW, Article 72 (Dec. 2017), 21 pages. https://doi.org/10.1145/3134707

[9] Natalie Chiang. *"Natalie Chiang on Her Transition into Computer Science and Defying the Developer Stereotype"*. Retrieved October 15, 2020 from https://www.genw.ca/blog/natalie-chiang-on-her-transition-into-computer-scienceand-not-fitting-the-mould-of-a-software-developer.

[10] Saron Yitbarek. [n.d.]. *"Code Newbies Podcast"*. Retrieved October 15, 2020 from https://www.codenewbie.org/podcast.

[11] Whittaker, S., D. Frohlich, and O. Daly-Jones. *"Informal workplace communication: What is it like and how might we support it?"* In Proceedings of the ACM conference on human factors in computing systems (CHI'94) (Boston, MA, April 24, 1994). ACM Press, 1994, 131-137.

[12] Hudson, J.M., J. Christensen, W.A. Kellogg, and T. Erickson. *"I'd be overwhelmed, "*: Availability and interruption in research management. In Human factors in computing systems: Proceedings of CHI'02 (Minneapolis, MN, April 20, 2002). ACM Press, 2002, 97-104.

[13] Gonzalez, V.M. and G. Mark. *"Constant, constant, multi-tasking craziness": Managing multiple working spheres"*. In Human factors in computing systems: Proceedings of CHI'04 (Vienna, Austria, April 24, 2004). ACM Press, 2004, 113- 120.

[14] Czerwinski, M., E. Horvitz, and S. Wilhite. *"A diary study of task switching and interruptions"*. In Human factors in computing systems: Proceedings of CHI'04 (Vienna, Austria, April 24, 2004). ACM Press, 2004, 175-182.

[15] Bannon, L., A. Cypher, S. Greenspan, and M.L. Monty. *"Evaluation and analysis of users' activity organization"*. In Proceedings of the ACM conference on human factors in computing systems (CHI'83) (Boston, MA, December 12, 1983). ACM Press, 1983, 54-57.

[16] Faraj, Samer, and Lee Sproull. *"Coordinating Expertise in Software Development Teams"*. Management Science, vol. 46, no. 12, 2000, pp. 1554–68. JSTOR, http://www.jstor.org/stable/2661533. Accessed 4 Aug. 2022.

[17] Vinekar, V., Slinkman, C.W., Nerur, S., 2006. *"Can agile and traditional systems development approaches coexist? an ambidextrous view"*. Inf. Syst. Manage. 23 (3), 31–42. doi:10.1201/1078.10580530/46108.23.3.20060601/93705.4.

[18] Stray, V., Sjøberg, D.I.K., Dybå, T., 2016. *"The daily stand-up meeting: a grounded theory study"*. J. Syst. Softw. 114, 101–124. doi:10.1016/j.jss.2016.01.004

[19] Felstead, A. and G. Henseke (2017), *"Assessing the Growth of Remote Working and Its Consequences for Effort, Well being and Work life Balance"*, New Technology, Work and Employment 32 , 195–212

[20] Gloria Mark, *"Professor in the Department of Informatics at the University of California"* https://www.fastcompany.com/944128/worker-interrupted-cost-task-switching

[21] Souti Chattopadhyay, Thomas Zimmermann, and Denae Ford. 2021. *"Reel life vs. real life: how software developers share their daily life through vlogs"*. Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Association for Computing Machinery, New York, NY, USA, 404–415. https://doi.org/10.1145/3468264.3468599

[22] V.G. Stray, N.B. Moe, and T. Dybå, *"Escalation of Commitment:A Longitudinal Case Study of Daily Meetings, Agile Processes in Software Engineering and Extreme Programming"*. 13th International Conference, Springer, 2012, pp. 153-167

[23] V.G. Stray, N.B. Moe, and T. Dingsøyr, *"Challenges to Teamwork: A Multiple Case Study of Two Agile Teams"* Agile Processes in Software Engineering and Extreme Programming:13th International Conference, Springer, 2011, pp. 146-161.

[24] K. Schwaber and M. Beedle, Agile Software Development withScrum, Prentice Hall, 2001.

[25] J. Yip, *"It's Not Just Standing Up: Patterns of Daily Stand-upMeetings"*. http://www.martinfowler.com/articles/itsNotJustStandingUp.html,2011, pp. 1-17.

[26] https://blog.authenticjourneys.info/2020/09/meaning-techie-words-non-techies.html

[27] C. Parnin and S. Rugaber, *"Resumption strategies for interrupted programming tasks"* Software Quality Journal, vol. 19, no. 1, pp. 5–34, 2011.

[28] Joshi, K.D., Sarker, S., and Sarker, S. *"KnowledgeTransfer within Information Systems DevelopmentTeams: Examining the Role of Knowledge SourceAttributes"*. Decision Support Systems, Volume 43,Number 1, 2007, pp.322–335.

[29] Hewitt, B. and Walz, D. *"Using Shared Leadershipto Foster Knowledge Sharing in Information Sys-tems Development Projects"*. Proceedings of the 38th Hawaii International Conference on System-Sciences, 2005, pp.

[30] Dybå, T., Dingsøyr, T.: *"Empirical Studies of Agile Software Development: A Systematic Review"*. Information and Software Technology 50, 833–859 (2008)

[31] K. Schwaber and J. Sutherland, *"The Scrum Guide."* http://www.scrum.org/scrumguides/ Scrum Allience, 2011.

[32] VersionOne, *"State of agile survey"*. http://www.versionone.com/state-of-agile-development-survey/11/2011.

[33] T. L. Dickinson and R. M. McIntyre, *"A Concetual Framework for Teamwork Measurement "*in Team Performance Assessment and Measurement: Theory, Methods, and Applications, M. T. Brannick, E. Salas, and C. Prince, Eds., ed, 1997, pp. 19-43.

[34] E. Salas, D. E. Sims, and C. S. Burke, *"Is there a "big five" in teamwork?,"* Small Group Research, vol. 36, pp. 555-599, Oct 2005.

[35] F.R.H. Zijlstra, R.A. Roe, A.B. Leonora, and I. Krediet, *"Temporalfactors in mental work: Effects of interrupted activities,"* Journal ofOccupational and Organizational Psychology, vol. 72, no. 2, pp.163-185, 1999.

[36] N. B. Moe, T. Dingsyr, and T. Dyba, *""A teamwork model for understanding an agile team: A case study of a Scrum project,"* Information and Software Technology, vol. 52, pp. 480-491, 2010.

[37] A. Luong and S.G. Rogelberg, *"Meetings and More Meetings: TheRelationship Between Meeting Load and the Daily Well-Being ofEmployees,"* Group Dynamics: Theory, Research, and Practice,vol. 9, no. 1, pp. 58-67, 2005.

[38] A.C. Bluedorn, D.B. Turban, and M.S. Love, *"The Effects ofStand-Up and Sit-Down Meeting Formats on Meeting Outcomes,"* Journal of Applied Psychology, vol. 84, no. 2, pp. 277-285, 1999.

[39] N.C. Romano Jr and J.F. Nunamaker Jr, *"Meeting analysis:Findings from research and practice,"* in Proceedings of the 34thAnnual Hawaii International Conference on System Sciences,IEEE, 2001, pp. 13.

[40] S. Kauffeld and N. Lehmann-Willenbrock, *"Meetings MatterEffects of Team Meetings on Team and Organizational Success,"* Small Group Research, vol. 43, no. 2, pp. 130-158, 2012.

[41] G. Asproni, *"Motivation, Teamwork, and Agile Development,"* Agile Times, vol. 4, pp. 1-9, 2004.

[42] S.G. Rogelberg, D.J. Leach, P.B. Warr, and J.L. Burnfield, *"Not Another Meeting! Are Meeting Time Demands Related toEmployee Well-Being?,"* Journal of Applied Psychology, vol. 91,no. 1, pp. 83-96, 2006.

[43] D.E. Perry, N. Staudenmayer, L.G. Votta, *"People, organizations, and process improvement"*, IEEE Softw. 11 (4) (1994) 36–45.

[44] S. Datta, R. Sindhgatta, B. Sengupta, *"Evolution of developer collaboration on the jazz platform: a study of a large scale agile project"*, Proceedings of the 4th India Software Engineering Conference, ACM: Thiruvananthapuram, Kerala, India, 2011, pp. 21–30.

[45] T. Zimmermann, N. Nagappan, *"Predicting defects using network analysis on dependency graphs"*, Proceedings of the 30th International Conference on Software Engineering, ACM: Leipzig, Germany, 2008, pp. 531–540.

[46] R.Abreu,R.Premraj, *"How developer communication frequency relates to bug introducing changes"*. Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops IWPSEE Vol 09, ACM: Amsterdam, The Netherlands, 2009, pp. 153–158.

[47] C. Bird, et al., *"Mining email social networks"*, Proceedings of the 2006 International Workshop on Mining Software Repositories, ACM: Shanghai, China, 2006, pp. 137–143.

[48] M. Cataldo, et al.,*" Identification of coordination requirements: implications for the Design of collaboration and awareness tool sv"*, 20th

Anniversary Conference on Computer Supported Cooperative Work, 06, ACM: Banff, Alberta, Canada, 2006, pp. 353–362.

[49] E. Shihab, et al., *"On the central role of mailing lists in open source projects: an exploratory study"*, New Frontiers in Artificial Intelligence, Springer, Berlin: Heidelberg, 2010, pp. 91–103.

[50] https://www.youtube.com/watch?v=z7mIkruXgJQab_channel=SinghinUSA

[51] https://www.youtube.com/watch?v=JrvgXQPkwEwab_channel=PetiaDavidova

[52] https://www.youtube.com/watch?v=yUegdy0wNnIab_channel=azisseno

[53] https://www.youtube.com/watch?v=obd7CeJdtr8ab_channel=marsela

[54] https://www.youtube.com/watch?v=ySxrVgCm7D0ab_channel=KevinChen

[55] https://www.youtube.com/watch?v=mOd7bU4DhQ4ab_channel=TylerSchott

[56] https://www.youtube.com/watch?v=nTzjmSesD-4ab_channel=Shuya

[57] https://www.youtube.com/watch?v=VQdniD1f6lkab_channel=Handrew

[58] https://www.youtube.com/watch?v=EXCagTkMNy4ab_channel=PURIJIT

[59] https://www.youtube.com/watch?v=0mFOf2UcOuQab_channel=kagato0110

[60] https://www.youtube.com/watch?v=4MUrNpysYhUab_channel=AshleeY

[61] https://www.youtube.com/watch?v=5aK0SwOk5BAab_channel=AlexLiu

[62] https://www.youtube.com/watch?v=aj0sOcgWERMab_channel=KcOkolo

[63] https://www.youtube.com/watch?v=ZLRa1SjtpiQab_channel=marsela

[64] https://www.youtube.com/watch?v=Gtpfx5ZPIlwab_channel=IrisTabea

[65] https://www.youtube.com/watch?v=PE-7M6r7Ptcab_channel=HeyMarquece

[66] https://www.youtube.com/watch?v=7Mce1W5hBVIab_channel=DhritiChawla

[67] https://www.youtube.com/watch?v=6AWgwvJZpKcab_channel=SilverchipLtd

[68] https://www.youtube.com/watch?v=Qcr0rP4d-YIab_channel=TatumBair

[69] https://www.youtube.com/watch?v=7cxhqe3dWr4ab_channel=JapaneseJourney

[70] https://www.youtube.com/watch?v=47gRcB05S-sab_channel=VyTran

[71] https://www.youtube.com/watch?v=O62kAvoO9lMab_channel=sarahpan

[72] https://www.youtube.com/watch?v=oND980ijADoab_channel=CSJackie

[73] https://www.youtube.com/watch?v=1ib-H2iAuq4ab_channel=LeonZhu

[74] https://www.youtube.com/watch?v=jhSMg-AW35Mab_channel=ChildishJordino

[75] https://www.youtube.com/watch?v=5ELQoxVKMsQab_channel=MansoorCodes

[76] https://www.youtube.com/watch?v=nTzjmSesD-4ab_channel=Shuyai

[77] https://www.youtube.com/watch?v=d4fKCZH-1HIab_channel=joernies

[78] https://www.youtube.com/watch?v=vt79JcPfZQAab_channel=
ConnectEd

[79] https://www.youtube.com/watch?v=BxA9slRVRfwab_channel=
GyasiLinje

A. Appendix 1 - Vlog List

No	Location	Gender	Age	Job Type	Position	Video Date (Month and year)
V1	USA	Male	24	Office	Developer/ Scrum Master	12/2021
V2	London	Female	25	Office	Software Engineer	12/2021
V3	Jakarta	Male	27	Office	Software Engineer	05/2018
V4	Los Angeles	Female	26	WFH	Software Engineer	10/2021
V5	USA	Male	31	WFH	Hybrid Software Engineer	01/2022
V6	Atlanta	Male	27	Office	Software Developer Intern	11/2021
V7	Tokyo	Male	30	Office	Software Engineer	12/2021
V8	New York	Male		Office	Software Engineer	11/2021
V9	Tokyo	Male	28	Office	Software Engineer	10/2020
V10	Japan	Male	32	Remote	Software Engineer	03/2020
V11	California	Female	30	Office	Software Developer Intern	10/2021

V12	London	Male	24	WFH	Software Engineer	05/2021
V13	London	Male	32	Remote	Software Engineer	03/2021
V14	Los Angeles	Female	28	WFH	Software Engineer	01/2021
V15	Spain	Female	28	Remote	Software Engineer	11/2021
V16	USA	Male	29	Office	Software Engineer	03/2020
V17	San Francisco	Female	27	WFH	Software Engineer	01/2022
V18	Manchester, UK	Male	30	WFH	Software Developer	02/2022
V19	New York, USA	Female	25	WFH	Associate Software Engineer	04/2021
V20	Japan	Male	28	WFH	Software Engineer	02/2021
V21	Chicago	Female	26	WFH	Software Developer Intern	08/2021
V22	New York, USA	Female	27	WFH	Software Engineer	01/2022
V23	London	Female	24	WFH	Software Engineer	01/2022
V24	Canada	Male	25	WFH	Software Developer Intern	01/2021
V25	Michigan	Male	30	Office	Software Engineer	10/2019
V26	Canada	Male	27	WFH	Software Engineer	04/2021
V27	Tokyo	Male	26	Office	Software Engineer	12/2021

V28	Los Angeles	Male	28	WFH	Software Engineer	06/2022
V29	USA	Female	27	WFH	Software Engineer	03/2015
V30	New York	Male	29	Remote	Frontend Engineer	08/2021

Table A.2.: Vlog list

A. Appendix 2 - Vlog Analysis

No.	Coding and programming	Meetings	Remote Work/WFH
V1	Presentation of work	Daily Standup, KT Meeting	
V2	Code blurred, Pull request and testing,Fixing bugs, documentation, Epic (feature) Planning, Timelines	Daily Standup, Weekly meeting (2 hours)	
V3	KT in team, Pair programming	Team discussion, team meeting	
V4	Blur window, pull request Code review, releasing a new version, bug fixing, (Not much time for coding due to meeting)	Standup meeting, company-wide meeting (2 hours) , Team wide demo meeting, Retro meeting	Working a bit late, Flexible hours (attended meetings)
V5	Coding (8–12),Afternoon coding work	Meeting with the team (planning, discussion), to discuss issues going on with the project.	Attend meetings from home and then go to the office.
V6	Coding session with manager, API Backend in Python, meeting after lunch.	Standup meetings	

V7	Coding (when not in a meeting)	Standup meeting, team discussion, team meeting in the evening, One-on-one meeting (daily work sharing and communication tasks)	First, we'll meet at home, and then we'll go to the office.
V8	Code review, reading documents, coding, working on slack threads	Daily standups, coding interviews	
V9	Coding	Planning meetings	
V10	Coding, code review, bug fixing	Scrum meeting	
V11		team meeting,	
V12	Coding	Daily meeting	WFH due to COVID
V13	Review: Pull request, Jira board update, coding, C# , Testing	Team meeting	work from home.
V14	coding	Standup meeting, team meeting	
V15	Slack threads, Frontend and backend coding, Pair Programming, Java	One-to-one meeting, team meeting, meeting with mentor	Fully remote
V16	Code writing, code review in a team, JIRA	Meeting in teams	
V17	Android coding, testing, fixing bugs, cleaning up code, tickets done	Team Meeting, Flexible	Remote Work, Flexible working hours

V18	Coding, Work on assigned tickets, minor changes, bug fixes, new feature deployment	Daily Standup meeting, other team meetings	WFH due to pandemic, Flexible Working hours
V19	DevOps and backend development, unit testing, bug fixes 10-to-20 %-day coding, Pair Programming with the team	Meetings on a daily basis, with teammates, with newly joined groups, with the boss, and with other code	WFH, flexible working hours
V20		Team meeting	
V21	Android, bug fix, document designing, Backend development	2 hosts(guides), weekly update meeting, and a daily meeting with guides	Work from home due to pandemic
V22	Coding and writing tests	Stand-up meeting, PM and UX designer, One-on-one meeting: state of the project	
V23	Working on a ticket, documentation, and blurred screen	Every day stand-up, On-call meeting, Sprint planning, Team meeting	COVID pandemic
V24	Coding	Daily standup, meeting with the manager	Flexible working hours
V25	Coding, Pair programming, bug finding and fixing, KT in team	The daily standup meeting	
V26	Jira, Coding, Start with an easy task, API code, backend application	Daily Standup,Feature decision meeting	Home officeDue to Lockdown

V27	Coding	Standup meeting, team discussion, team meeting in the evening, One-on-one meeting	
V28	Building new features, fixing bugs, Swift UI, React Native, JavaScript	Standup meetings, product meetings, architecture meetings, architecture meetings	
V29	Website developer, backend, python, HTML, CSS, JavaScript	Team meeting, discussion in the team	
V30	Coding, training, starting a new story, working with React features, working on different features	There will be no meeting on Friday, but there will be some informal meetings with the team.	Full remote work

Table A.2.: Vlogs Analysis