Impact of technical debt on developers' morale and productivity

Bhuwan Paudel

Thesis submitted for the degree of

Master in Informatics: Programming and System Architecture

60 credits

Department of Informatics

Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Fall 2021

Impact of technical debt on developers' morale and productivity

Bhuwan Paudel

Abstract

Background: Technical debt (TD) is an emerging concept in software engineering. TD has been widely spoken from the technical, financial, and organizational perspectives. However, it is crucial to understand that there could be a considerable psychological cost beyond those economic and technical downfalls. So, this study explores the technical debt from human perspectives, specifically developers' morale and productivity.

Aim: This thesis aims to explore how the occurrence and management of TD impact the developers' morale and productivity. The impact of the occurrence of TD is studied based on the two aspects of TD, such as: 'introducing TD' and 'criticism after introducing TD.' Likewise, four aspects of TD, such as 'review,' 'repaying TD,' 'praise after repaying TD,' and 'collaboration to pay TD,' are considered to study the impact of management of TD on developers morale and productivity.

Methods: A qualitative research method was used to perform this research. Data were gathered by conducting semi-structured interviews with 11 software practitioners from different IT companies in Nepal and Norway. Thematic analysis using NVivo was performed to analyze the collected data.

Results: The results indicate that the occurrence of TD usually hurts both developers' morale and productivity. However, exceptions apply to a few developers whose morale and productivity do not get affected because of the presence of TD. Although repaying technical debt is a tedious effort that hinders developers' morale and productivity, other aspects of management of technical debt enhance productivity and morale. However, developers' satisfaction or motivation and productivity raise after successfully repaying technical debt.

Conclusion: This thesis concludes that TD, developers' morale, and productivity are co-relateable and should be studied together. The results show that apart from the financial and technical aspects, developers' morale is an essential human aspect of TD which can cost a higher interest in terms of reduced productivity in the future if not properly repaid. The introduction of TD and criticism for introducing TD both have negative impacts on morale and productivity. Team collaboration, the culture of praising each other, and review all contribute to managing TD, ultimately increasing the developers' morale and productivity. On the other hand, developers do not like to be a part of repaying TD because it has detrimental impacts on their morale and productivity.

Acknowledgment

The journey of writing a long master thesis has been challenging yet exciting for me. It would not have been possible to complete this work without the continuous support, guidance, and motivation from many people around me. Foremost, I would like to acknowledge and appreciate my two supervisors Assoc. Prof. Antonio Martini and Yngve Lindsjørn for their continuous effort, invaluable guidance, and encouragement throughout all stages of writing this thesis. Furthermore, I am incredibly thankful to all interviewees who participated in this study for the interviews. Without their enthusiastic participation, it would not have been possible to come out so far.

I would like to use this opportunity to express my gratitude to my family and friends for their encouragement and consistent support throughout my entire journey to the master's degree. A special thanks to my friend Sparsh, who guided me on writing at the early stages of this thesis and helped me in finding some interviewees.

Bhuwan Paudel
November, 2021

Table of Contents

Table of Figures

List of Tables

1. Introduction

1.1 Technical debt, developers' morale, and productivity

Ward Cunningham (Cunningham, 1992) used the term "technical debt" for the first time in software development two decades ago to convey the need for what we now call "refactoring" to nontechnical product stakeholders. Since then, it has been improved and expanded widely (Kruchten, Nord, & Ozkaya, 2012). Cunningham used the term TD to describe the potential long-term detrimental consequences of immature code written during the software development process. TD is a sub-optimal, limited, or relatively easy technological solution instead of better approaches available that results in a short-term advantage and the future payment of Interest (Ghanbari, Besker, Martini, & Bosch, 2017).

In recent years, technical debt has been one of the most emerging research topics within the software engineering field. However, those studies are mainly focused on the technical, financial, and organizational aspects of TD (Ampatzoglou, Ampatzoglou, Chatzigeorgiou, & Avgeriou, 2015), (Li, Avgeriou, & Liang, 2015). (Becker, Walker, & McCord, 2017) argues that little attention has been paid to the impact of TD on human aspects. Even though (Besker, Ghanbari, Martini, & Bosch, 2020) have performed an empirical investigation to define the relationship between technical debt, developers' morale, and productivity, exploring the impact on developers' morale and productivity independently based on the more specific key aspects related to TD (like introducing and repaying TD, criticism for introducing TD, praise after repaying TD, review, team collaboration for repaying TD) will be beneficial to strengthen the previous results and can lead to the new outcomes.

(Tom, Aurum, & Vidgen, 2013) says that repaying technical debt is a tedious and frustrating task for many developers, so, in the long run, developers might feel disappointed with the decision of introducing technical debt. Likewise, interviewees in a study (Yli-Huumo, Maglyas, & Smolander, 2014) remarked that introducing technical debt is not a good feeling for developers. On the other end, sometimes, allowing technical debts helps some developers to set a good working environment where they can spend their full time on development, deploying the new product to the market, or adding the new feature to the existing one. In this sense, technical debt can enhance those developers' short-term morale and as well as performance (Tom et al., 2013). Unfortunately, when it comes to the developers who are very serious about quality and productivity, it has strong negative influences, it hurts developers'

morale even in the short term (Tom et al., 2013). (Olsson, Risfelt, Besker, Martini, & Torkar, 2020) found that dealing with TD is unpleasant and may require particular team expertise to be effectively managed.

Different exploratory studies have concluded that happiness and positive morale are directly proportional to performance and productivity (Graziotin, Fagerholm, Wang, & Abrahamsson, 2018). A few numbers of previous studies (Ghanbari et al., 2017), (Besker et al., 2020), (Olsson et al., 2020), (Besker, Martini, & Bosch, 2019), (Tom et al., 2013), (Li et al., 2015) argue that occurrence and management of technical debt, developers' morale, and productivity are correlated. Notwithstanding, the actual relationship among these factors still lacks more empirical evidence (Besker et al., 2020).

Although one of the latest studies (Besker et al., 2020) addressed how technical debt's occurrence and management influence developers' morale, there is still scarce evidence on how a specific aspect related to the occurrence and management of technical debt affects developers' morale and productivity. The study (Besker et al., 2020) clearly highlighted that the occurrence of TD has a negative influence, whereas management has solid positive influences on developers' morale. These two statements need more empirical evidence because the occurrence of TD could have at least a few positive or no impacts on morale, whereas the management of TD could have adverse effects. This is because introducing technical debt can yield short-term benefits (Li et al., 2015) in fact, increase short-term morale (Tom et al., 2013). On the other hand, repaying TD (an essential aspect of managing technical debt) is a stressful and frustrating effort for developers (Tom et al., 2013). It means not all the aspects of TD management necessarily have positive impacts on developers' morale and productivity all the time.

So, this study intends to find the relationship among technical debts, developers' morale, and productivity based on specific aspects related to the occurrence and management of TD. Additionally, provide more evidence to the findings of a few previous research in this field such as, (Besker et al., 2020), (Olsson et al., 2020), (Ghanbari et al., 2017), (Spínola, Vetrò, Zazworka, Seaman, & Shull, 2013), (Tom et al., 2013), (Yli-Huumo et al., 2014), and other relevant studies. The impact of the occurrence of TD will be studied based on two aspects, such as 'introducing TD' and 'criticism after introducing TD.' In contrast, impacts of management of TD will be analyzed based on four aspects: 'review,' 'repaying TD,' 'praise after repaying TD,' and 'team collaboration for repaying TD.' Studying these individual TD aspects has

produced exciting, encouraging, and in-depth results, providing more insights into TD's impacts on developers' morale and productivity. The findings of this study support the previous outcomes, but in some cases, contradict them as well, which are well described in section 5. The specific aspects of TD as mentioned above and their impacts on developers' morale and productivity are described in detail in section 2.4.2, 2,4,3, and 2.4.4 respectively.

1.2. Motivation

In recent years, as software has advanced at such a breakneck pace, the concept of technical debt has gained close attention as a prominent concept in software engineering research (Brown et al., 2010). After reviewing a few studies, I realized that technical debt is one of the fastest-growing and most important research areas, with tremendous future potential in the age of cutting-edge technology. However, human aspects of technical debt continue to lack enough research and quantifiable findings. As a result, I chose to focus my master's thesis on this area of study.

I was partially familiar with the concept of technical debt in a lecture on IN5140: Smart process and agile methods during my first semester. I found that concept fascinating and went through more literature. After that, I acknowledged that researchers are mainly looking at the economic, technological, and organizational elements of technical debt. As a result, there is a paucity of studies and findings on software developers' morale or psychological aspects. The involvement of individual developers in the occurrence of TD and the repercussions of the TD for individuals has received less consideration in the prior literature (Ghanbari et al., 2017). So, this research gap between technical debt and its impact on the human aspect prompted me to select this research topic for my master's thesis. In addition, to the best of my knowledge, the software industry still needs more evidence on how developers' morale (affected by the technical debt) impacts their productivity.

Even though a few exploratory studies (Besker et al., 2020), (Olsson et al., 2020), (Ghanbari et al., 2017) particularly investigated the core relationship among the TD, developers' morale, psychological consequences, and productivity, some specific key aspects of TD such as introducing and repaying TD, team collaboration, review, criticism or praise and their impacts on developers' morale and productivity need to be studied in details to comprehend their results. So, the main motivation for this study came after knowing that in addition to specific technical and economic ramifications, TD has some adverse effects on developers' morale and productivity. Studying the influence of more specific aspects of TD on developers' morale and productivity and comparing the result with previous studies would be interesting and helpful to extend the results. Thus, this study is motivated by a willingness to support the earlier findings and also investigate novel outcomes in the research field of TD and impact on its' human aspects (particularly developers' morale and productivity).

1.3 Research Questions

"A research question is a question that a research project sets out to answer (Mattick, Johnston, & de la Croix, 2018)*."* The most important and arguably most challenging component of a study design is generating research questions in any research. Establishing good research questions is crucial because research design, methods, and results are all supposed to be made to answer research questions (Bryman, 2007).

As mentioned earlier in the previous section, the relation between technical debt and its' human aspect (developers' morale and productivity) needs more exploration and research. Thus, this study is mainly focused on finding the impact of technical debt on developers' morale and the influence made by developers' morale on their productivity via occurrence and management of TD by exploring the following research questions:

RQ1: How do occurrence and management of technical debt affect the developers' morale?

Rationale: RQ1 aims to define the relationship between TD aspects and developers' morale. Here, the occurrence of TD includes two aspects, such as introducing TD and getting criticism after introducing TD, whereas management of TD comprises four aspects, such as like reviews, repaying TD, getting praise after repaying TD, and team collaboration. So, answering RQ1 may persuade the software practitioners and researchers to consider the impacts of different aspects of TD on morale more seriously. Additionally, RQ1 explores the impacts of occurrence and management of TD on developers' morale and knowing those possible impacts earlier could let software practitioners make wise decisions while taking and managing technical debt. For instance, if this study finds that introducing TD hinders developers' morale, software practitioners could think twice before compromising quality standards.

RQ2: How does developers' morale affect their productivity via the occurrence and management of TD?

Rationale:

Here, morale refers to specific developers' morale which was changed because of the occurrence and management of TD. RQ2 aims to investigate how the developers' productivity is affected when developers' morale is impacted. The answer to this research question helps developers' to understand that occurrence and management of TD affects developer morale which is directly related to their productivity. Additionally, software

practitioners will know how important morale is to increase productivity. So, answering RQ2 could help software companies and developers to improve their productivity by performing activities that boost their morale while dealing with the occurrence and management of TD.

1.4 Approach

Qualitative interviews were performed with different software professionals from different software companies in Norway and Nepal to answer these two research questions. The interview had different types of questions that qualitative interviews should have. For instance, interviewees were asked to share their feeling after introducing or repaying technical debts. How does criticism after introducing TD, praise for repaying TD, or review process affect their morale?

The same kinds of questions were asked to know how developers' morale affects their productivity. All the details about research design, methods, data collection, and data analysis to answer these research question is presented in chapter 3.

1.5 Structure of Thesis

Chapter 2: Background and related works highlight a brief introduction to technical debt, different types and management practices of TD, risk of technical debt on morale and productivity. Most importantly, human aspects of technical debt are explained in this chapter which includes the impact of technical debt on developers' morale and productivity.

Chapter 3: Research methodologies mainly describe the research method, research design, preliminary studies, data collection methods, and data analysis approach used in this study.

Chapter 4: Description of the studied interviewees portrays a detailed explanation of each interviewee studied, including general information like current role, experience, and education.

Chapter 5: Results depict the outcome from data analysis in connection with the background study presented in 2.6: human aspects related to technical debt.

Chapter 6: Discussions describes the results, compare with related work, and answer the research questions

Chapter 7: Conclusion and future work present the conclusion of the study focused on research questions and potential future work in this research area.

2. Background and Related works

This chapter describes the background information, previous works, and research relevant to this thesis. Firstly, the concept of technical debt, including its' impact on developers' morale and productivity, will be described since it is related to the research context. Then, I will present a conceptual framework of this study that explores different aspects of TD that affect developers' morale and productivity.

2.1 Concept of Technical Debt

In general language, the term "debt" is defined as a sum of the amount that is owed or due, which is precisely co-relatable in software engineering as well. Debt always has two sides. Sometimes it improves welfare and generates higher revenues. However, sometimes it can be catastrophic if it is used indiscreetly and in abundance (Cecchetti, Mohanty, & Zampolli, 2011).

The conclusion of a seminar presented by (Avgeriou, Kruchten, Ozkaya, & Seaman, 2016) defined the term technical debt as: "*In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability* (Avgeriou et al., 2016)." (Brown et al., 2010) characterize the concept of TD as the gap between the potential outcomes of software systems at the present state and the ideal state in which the system is ideally effective in a specific environment. The known flaws, unimplemented features, and out-of-date documentation are all examples of debt (Brown et al., 2010).

In this era of cutting-edge technologies, the growth of technological advancement is immense, and the competition among different companies, IT professionals to sell their software products and services is rapidly increasing. A new trend of releasing products early to market than their competitors has pushed the growing software companies to accelerate the latest releases of their valued products onto the hands of their users as soon as possible (Yli-Huumo et al., 2014). This may help to earn additional benefits initially, but such benefits can cause higher costs in the future. In the midst of all of this, the developer is being tasked with more work than ever before, resulting in decreased maintainability (Yli-Huumo et al., 2014), (Li et al., 2015).

Technical debt accumulated throughout the software development process may increase developer productivity in the near term, but debt eventually outweighs the overall gain achieved. In the long run, the debt may have to be repaid or refactored with interest, and the interest is the adverse effects in terms of additional works and efforts that must be paid as a result of accumulated TD (Besker et al., 2019). Although accumulating TD may result in a gain between the short-term benefit and the interest paid, in many circumstances, as documented in literature and software projects, the interest may far outweigh the gain, for example, by causing development crises (Ghanbari et al., 2017).

(Allman, 2012) defined TD from a different perspective that not all technical debts are wrong or harmful to the software and team. TD can also be incurred intentionally on purpose to gain short time benefits such as saving time, money, or other resources. Sometimes, a software team or developer might consciously want to accumulate a TD to achieve some short-term benefits and get a leg up on the competition. In some situations, accumulating TD may be unavoidable, and it might even be beneficial if properly managed. However, at the same time, this can be tricky as TD emerges from a variety of factors, making it impossible to foretell the effects and usually involves a gamble on what will happen in the future. So, TD might turn out as a bad gamble or cause unplanned maintenance and missing deadlines (Allman, 2012).

Cunningham (Cunningham, 1992) argues that the source code with possible future technical debts could lead the development of products in the shortest possible time, which works fine and is acceptable to the customer initially. However, in the long term, this process will make the overall system more complex and complicated results an inflexible product. Cunningham says that "*Shipping first-time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite"* (Cunningham, 1992, p. 29). Many negative impacts and effects can arise if those debts are not paid with interest on time (Cunningham, 1992). In the field of the software development industry, technical debts are mostly considered as the critical issues because the extra bill caused by those debts is very high. The concept of financial debts depicts the nature of increasing software development costs over time (Tom et al., 2013).

2.2 Debts, Interest, and principal.

The term debt is defined as "*the sub-optimal solution implemented to achieve short-term benefits*" (Besker et al., 2020). Technical debt (TD) is a suboptimal solution in code (or other software development artifacts) that provides a short-term gain but incurs additional long-term costs over the software's lifecycle (Besker et al., 2020).

A fundamental characteristic of technical debt in all its form is the concept of repaying interests (Tom et al., 2013). The interest of the technical debt is "*the additional effort that is needed to be spent on maintaining the software, because of its decayed design-time quality*" (Ampatzoglou et al., 2015, p. 63). Tom et al. (Tom et al., 2013) argue that interest payments on technical debt are connected with negative effects on morale, productivity, and quality. Reduced productivity is an interest expense in and of itself, potentially resulting in more debt.

Besker et al. (Besker et al., 2020) define the term principal as "*the cost of refactoring the sub-optimal solution."* To put it another way, the principal of a technical debt item is the amount that would have to be paid off in order to remove the debt item. This is an effort necessary to eliminate a technical debt item by changing the software (Fernández-Sánchez, Garbajosa, & Yagüe, 2015).

2.3 Risk of technical debts on developers' morale, productivity, and software quality

Developers' morale

In the long run, developers are compelled to repay interest and principal on accrued debt in order to sustain and modify their system. As a result, technical debt has a long-term detrimental effect on morale (Tom et al., 2013). Likewise, a study (Fernández-Sánchez et al., 2015) has shown that technical debt negatively influences software development, including implications on developers' morale, team velocity, and product quality. Technical debt, for example, can lead to low morale within development teams, resulting in systemic issues such as developer turnover.

Software quality

The result of the study (Tom et al., 2013) finds that when trade-offs that introduce technical debt must be made, certain quality attributes such as extensibility, scalability, maintainability, adaptability, performance, usability, testability, supportability, reliability, and security were compromised. So, this finding indicates that developers usually compromise software quality while introducing technical debt. Technical debt adversely affects the quality of products in terms of flaws and other structural concerns. Technical debt in any form degrades quality in the short term, but when it accumulates over time, the quality degrades more as a result of interest expenses. Technical debt contributes to the formation of flaws and the fact that defects in a product go unreported (Tom et al., 2013).

Developers' productivity

Another risk of introducing technical debt is strongly intertwined with its impact on productivity. Technical debts can create a high level of uncertainty about making any changes because developers might be unable to identify what and where it is wrong. Even the best developers find it challenging to estimate. So, this uncertainty increases the chance of late deliveries, ultimately reducing productivity (Tom et al., 2013). According to the authors, because the occurrence of TD degrades software quality, developers must invest additional time and effort in addressing quality issues in the future. This, in turn, will result in a long-term drop in developers' productivity and maintenance expenses (Fernández-Sánchez et al., 2015), (Tom et al., 2013), (Besker et al., 2020).

2.4 Human aspects related to technical debt

Software development is a human-based, socio-technical phenomenon and a psychological activity (Besker et al., 2020), (Fagerholm & Münch, 2012). The research study (Fagerholm & Münch, 2012) explains that the success of any software development process highly depends on human factors. To maintain the quality and increase the rate of performance or productivity, technical aspects along with human and social factors play a vital role. The first important thing that software developers should have is self-motivation because the overall development process requires better and more exhaustive comprehension of developers' feelings, perceptions, inspirations, and identification with their tasks in their respective project environments (Fagerholm & Münch, 2012).

Some essential disciplines such as positive affective states, happiness, satisfaction, and motivation increase software developers' productivity and software quality. The finding of research (Graziotin et al., 2018) explains that "*developers experience happiness as a productivity and performance booster, and unhappiness has the opposite effect*: *productivity and performance are the aspects which suffer most from unhappy developers*."

Further, literature like (Tom et al., 2013), (Li et al., 2015), (Yli-Huumo et al., 2014), (Spínola et al., 2013), (Olsson et al., 2020; Ozkaya, 2019), (Damian & Chisan, 2006), (Laribee, 2009), (Hardy, 2010) have indicated that the occurrence and management of technical debt somehow affect the human aspects specifically developers' happiness, psychology, morale, and productivity. Moreover, most of these studies indicate that technical debt has negatively impacted software practitioners' morale and productivity. A detailed study on the impact of technical debt on developers' morale, happiness, and productivity will be discussed in the following chapters.

2.4.1 Morale

Team Morale: "*A cognitive, emotional, and motivational stance toward the goals and tasks of a group. It subsumes confidence, optimism, enthusiasm, and loyalty as well as a sense of common purpose*" (Peterson, Park, & Sweeney, 2008). Morale was initially a topic of interest and used to be practiced in the military, and non-scientific terms like 'spirit' and 'soul' were used to define morale(Hardy, 2010). Consequently, so many organizations paid attention to this research topic after the second world war. Different definitions such as satisfaction, motivation, happiness, cohesion, group goals, and enthusiasm have been used to describe

morale. However, they have their distinct meaning (Besker et al., 2020), (Guion, 1958). Apart from these definitions, some other factors like teams, the relationship among the team members, goals, and objectives are also important aspects of morale (Hardy, 2010). Guion defines: *"Morale is the extent to which an individual's needs are satisfied and the extent to which the individual perceives that satisfaction as stemming from his total job situation."* (Guion, 1958, p. 62). Likewise, Guba (Guba, 1958, p. 198) defines moral as *"A predisposition on the part of persons engaged in an enterprise to put forth extra effort in the achievement of group goals or objectives*." Zeitz (Zeitz, 1983, p. 1098) uses the definition that *"Morale concerns members' affective or emotive responses to the organization-their general sense of well-being and enthusiasm for collective endeavors*."

Morale is a very subjective phenomenon; it is difficult to define in particular words or sentences. Hardy (Hardy, 2010) suggests that we can predict morale by measuring a set of factors that influence morale, and those antecedents factors are differentiated into three dimensions: affective, future/goal, and interpersonal.

The affective antecedents are related to the individuals' thoughts and their personal feelings, instant moods, individuals' sense of self-worth, emotions, attitudes, and their sense of identity (Hardy, 2010), (Besker et al., 2020). Positive affective antecedents such as praise, recognition, good feedback, sense of achievement, or happiness are the most important factors to attain high morale. While, negative affective antecedents like criticism, a sense of injustice, or the absence of praise are responsible for generating low morale (Hardy, 2010).

"The belief that tomorrow will be better than today is a critical factor in improving morale"(Hardy, 2010). The second dimension: future/goal antecedents are related to personal perceptions or feelings about the differences between today and from the future perspective, which can significantly affect morale. The combination of factors like a goal, sense of progress towards achieving that goal, belief in a better future, or success can increase the morale, while some factors like lack of confidence/clarity, no progress, no means of encouragement lower morale (Hardy, 2010).

Thirdly, the interpersonal antecedents of morale reflect the personal relationship, communication, and networking between individuals (primarily coworkers). Examples of antecedents of high morale are: Improving an existing interpersonal relationship, expanding their network within the organization, and team collaboration. On the other end, lack of

proper communications, absence or fragmentation of personal relationships, bullying, or being dragged down by coworkers can reduce morale (Hardy, 2010), (Besker et al., 2020).

2.4.2 The relationship between occurrences of technical debt and developers' morale

Introducing or the presence of TD and its impacts on developers' morale

Most of the existing literature and researches on this area mostly prioritized the technical and economic aspects of TD. However, an analysis performed by J. Olsson and co (Olsson et al., 2020) explores from a different perspective (i.e., human aspect). Their study suggests that morale is one of the most important aspects to consider when discussing human aspects in TD research. Online and paper surveys with different software professionals performed in a research article (Spínola et al., 2013) find that working without debt can be a solid motivation to achieve high morale while exiting of TD negatively impacts. A table (Besker et al., 2020, p. 4) mentioning a list of previous studies illustrates the relationship between technical debt and developers' morale. Most of the literature listed there in the table (Besker et al., 2020, p. 4) has suggested that the occurrence of TD has negative impacts on developers' morale.

In the interviews conducted by a research study (Ghanbari et al., 2017), several developers have shared their experience after realizing that they had introduced TD as negative feelings. They described those negative feelings in terms of insecurity, failures, or being afraid and upset. A qualitative analysis performed in the research (Olsson et al., 2020) finds that a high amount of TD affects the practitioners' state of mind, bringing anxiety among them. As a consequence, massive loss in their productivity. (Yli-Huumo et al., 2014) also suggests that taking technical debt is not a good feeling for developers (Besker et al., 2020). According to (Besker et al., 2020), the presence of TD is linked to a lack of progress and a waste of time, and this could have a destructive impact on the morale of developers. Unmanaged code will lead to chaotic behavior and annoyance, limit understandability, and, as a result, induce stress in its developers, as well as a never-ending cycle of defects, vulnerabilities, and technical debt. As a result, developer morale will eventually suffer and go down. Hence, product quality is a function of not just delivering products that match customer expectations but also of maintaining a healthy development organization that reflects the developers' voice (Ozkaya, 2019).

The study (Aldaeej, 2019) discovered a causal relationship between early to market strategy of software development and accumulation of TD. It shows that startups are incurring so many technical debts as an investment to develop the software faster, which may never be repaid. According to the study (Aldaeej, 2019), TD has detrimental effects on morale, productivity, and product quality in startups. Likewise, the result of the study by (Besker et al., 2020) finds that when developers believe that TD is impeding their work, their morale suffers and declines. As a result, they start to judge and rate their achievement lower and the waste higher. In other words, developers with low morale appear to have an exaggerated sense of how much TD obstructs their work and how much time they waste managing TD. This, in turn, may lead to a poor self-evaluation (i.e., the developer encounters more waste), which lowers their morale even further, thus worsening their view of the time wasted due to TD (i.e., negative self-assessment) (Besker et al., 2020).

The study (Tom et al., 2013) argues that technical debt's short-term impact varies depending on the individual choices and the circumstances. Developers working continuously on implementing new features by allowing technical debt to accumulate (i.e., without paying attention to technical debt) can actually increase the short-term morale for some individuals. On the other hand, technical debt has a detrimental influence on morale, even in the short term, for developers who take the quality of the code they write very seriously (Tom et al., 2013).

Developers are obliged to repay interest and principal on accumulated debt in the long run so that they can continue to adapt and maintain their system. As a result, technical debt has a long-term negative impact on morale. Developers' morale degrades as technical debt continues to grow with each sub-optimal solution implemented without regard for breaking the vicious cycle of technical debt. This progressive kind of technical debt is particularly frustrating in the long run (Tom et al., 2013). For many developers', dealing with technical debt is a tedious effort, and the consequences of technical debt are likely to be annoying and frustrating in the long run. (Laribee, 2009) explains that *"beyond the obvious economic downside, there is a real psychological cost to technical debt. . . this psychological effect takes a toll on team morale".*

Criticism for introducing technical debt and its impact on morale

(Hardy, 2010) indicates criticism as a negative affective antecedent responsible for generating low morale, so it is worth studying the impact of criticism for introducing TD on developers' morale. Although most of the interviewees in the study (Besker et al., 2020) mention that they have not been criticized for introducing TD. They do not usually like to make decisions that lead to the occurrence of TD. However, it is still crucial to study this aspect related to TD to know the developers' opinion on how they will feel when criticized for introducing TD. The correlation between criticism and sense of progress discussed in (Besker et al., 2020) shows that developers usually get motivated to pay TD if they were not criticized for taking TD. Criticism could be in different forms for different purposes, but the criticism here refers to that particular criticism received only for introducing technical debt or violating quality standards.

2.4.3 The relationship between management of TD and developers' morale

Based on previous research, the study (Li et al., 2015) urge that technical debt negatively affect the maintainability of software systems. According to the same research, there are numerous methods for eliminating TD and mitigating its effects in existing projects and software teams. The management of TD entails several activities (also called TDM activities) that enable software development teams to prevent the occurrence of TD and deal with existing TD and its unwanted consequences (Ghanbari et al., 2017), (Li et al., 2015). So, this section discusses four activities (aspects of TD) related to managing TD and their impacts on developers' morale as follows:

Repaying TD and its impact on developers' morale

According to the findings of a study (Ghanbari et al., 2017), refactoring or repaying TD is an exciting task for developers since it both encourages them and enables them to continuously improve the quality of software products and feel secure about them. However, repaying technical debt is not an easy and straightforward task to perform. It might be problematic in some circumstances since it may result in system failure (Ghanbari et al., 2017). So, interviewees in the study (Ghanbari et al., 2017) believe that technical debt is an unavoidable aspect of the software development process that should be identified, documented, and repaid as a regular activity. Most importantly, the result indicates that developers feel satisfied and happy with repaying technical debt (Ghanbari et al., 2017).

Developers usually feel successful if they able to repay technical debts. Such feelings of achievement spread the positivity that enables developers to work efficiently and effectively (Besker et al., 2020). On the other side, the presence of technical debt has a negative effect on the developers' morale, whereas repaying technical debt has a beneficial impact on morale (Besker et al., 2020). Since repaying TD improves the software quality (Besker et al., 2020), such continuous enhancements boost developers' morale. They usually feel motivated, and their morale improves as a result of software improvement (Damian & Chisan, 2006). Working without debt may be motivating and beneficial in terms of achieving developers' morale. If working off TD improves team morale and motivation, then repaying TD may have a more significant positive effect on a software project (Spínola et al., 2013).

In contrast, a high amount of TD can create many negative consequences, and developers are responsible for solving them like a daily activity. Spending more time on repaying debts rather than investing in development makes developers more apathetic. In the long run, it harms the software practitioner's psychology (Olsson et al., 2020). The developers/engineers do not want to spend their time repaying debts, and most of the tasks related to repaying technical debt "are not very fun" (Tom et al., 2013). The result of an online survey (Evans Data Corp, 2018) shows that 76% of the developers reported that paying down TD hurts their morale (Besker et al., 2020, p. 4). However, the management of TD can positively impact developers' morale (Besker et al., 2020).

The review process and its impact on developer morales

The review in early-stage software development could help to maintain the quality standard by following the guidelines. Since reviews help to maintain the quality, it prevents developers from introducing technical debt. Even though the technical debt is introduced in the software, the review allows them to identify and manage them well. For example, code review can help to reduce the TD (Yli-Huumo et al., 2014). Studying the impact of reviews specifically on developers' morale and productivity could lead to interesting findings because the review is one of the interpersonal antecedents of morale (Hardy, 2010) related to the individuals' relationship and communication.

(Besker et al., 2020) argue that conducting reviews positively contributes to developers' skills and overall software quality, leading to satisfaction and motivation. Developers usually consider the review process (review from a colleague) as a positive contribution that allows them to identify potential TD and assist one another in improving software quality (Ghanbari

et al., 2017). Repaying, refactoring, or managing TD with the help of review enables developers to enhance their skills by understanding their mistakes and learning new aspects of software engineering. Since managing technical debt with the help of review would allow developers to improve the quality of software artifacts, which in turn boosts their confidence, morale, and, ultimately, performance (Ghanbari et al., 2017), (Besker et al., 2020).

The team collaboration and its impact on developers' morale

Team collaboration: Team collaboration integrates mutual communication and problem-solving abilities to meet a specific goal by encouraging people to work together (Hagerstownmarket.org, 2016). There have been numerous studies happening in recent times that reflect the importance of team collaboration enhancing the team morale of a software development team to manage the existing and possible technical debts. One of the results from a study performed by (Ghanbari et al., 2017) describes the management of technical debts as a collaborative effort in which all the team members contribute to a set of common goals within their teams.

One of the essential interpersonal antecedents of morale is team collaboration. A good interpersonal relationship with coworkers helps to achieve higher morale (Hardy, 2010). (Codabux & Williams, 2013) suggest that collaboration introduce a culture of sharing across the teams, which helps identify those who need extra support in the team. The use of daily stand-up meetings, burndown charts, communication reduces the TD at a lower cost (Shriver, 2010), (Tom et al., 2013). Repaying TD is a sense of progress, and positive feelings (Besker et al., 2020). Although (Ghanbari et al., 2017) argue that collaboration in a software team is crucial to managing TD and boosting morale and performance, it still lacks sufficient research on how developers feel when they do and do not get help from the team in repaying TD.

Furthermore, most participants in the same study (Ghanbari et al., 2017) believe that TD is an unavoidable phenomenon that must be recognized, documented, and repaid. They believe that good support, proper communication, and collective effort among all the members in a team to manage the TD is necessary for improving the quality of the software and their confidence as well. Hence, the study (Ghanbari et al., 2017) suggests that good support, proper communication, and collaboration among team members are crucial to overcoming TD and boosting morale and performance. Additionally, during the process of developing a software product, there could be involvement of developers from various modules. So, technical debt

could be inherited from one module to another while integrating work (Ghanbari et al., 2017). Majorities of interviews in the study (Ghanbari et al., 2017) stated that managing such kind of TD is a sense of progress towards the goals that can be accomplished by the collective effort of a team.

Praise for managing technical debt and its impact on developer morale

(Hardy, 2010) indicates that praise is another positive affective antecedent of morale to achieve high morale. Even though (Besker et al., 2020) indicated that praise is a code for high morale, a detailed study and more evidence is needed on its impacts on developers' morale and productivity. One of the correlations drawn from the data collected through an online survey (Besker et al., 2020) indicates that developers get encouraged and motivated when their team recognizes their efforts to identify and repay TD. So, showing appreciation, recognition, or praise is a good approach to motivate developers dealing with TD (Besker et al., 2020).

2.4.4 Developers' morale and its impacts on productivity via the occurrence and management of TD

In software engineering, productivity is mainly defined from the financial perspective; however, it does not yet have particular definitions. Productivity can also be measured in terms of efficiency and quality of output. Some constraints such as cost, schedule, scope, resources, and quality are considerable critical factors while measuring software productivity (Besker et al., 2019).

The study (Ghanbari et al., 2017) mentions that wasting time on repaying debts lowers the developers' morale down. According to a study by Tom et al. (Tom et al., 2013), the occurrence of technical debt harms not only morale but also quality and productivity. Since the TD reduces software quality, developers have to spend their work time on additional time-consuming activities. This directly makes an impact on their work progress; as a result, their performance reduces, and the cost of production increases. So, many studies suggested that developers do not want to work with TD because it lowers down their motivation toward the work (Ghanbari et al., 2017). Further, the management of TD increases developers' morale which somehow seems to increase the developers' productivity (Besker et al., 2019).

The result of the interview and survey conducted by (Ghanbari et al., 2017) with different IT practitioners working in the software industry shows that TD negatively impacts developers'

morale in terms of present and future goals but does not hamper the interpersonal relationship with other colleagues. In essence, the technical debt hinders the developers' efficiency in completing tasks, and accordingly, proper management of TD increases the developers' productivity (Ghanbari et al., 2017).

The survey performed by Graziotin (Graziotin, Fagerholm, Wang, & Abrahamsson, 2017) among different software professionals depicts that low code quality, coding practices, and inability to solve seen/unseen problems are the most influencing factors for developers' unhappiness. Likewise, developers' unhappiness as an adverse effect creates many consequences such as *low quality* and *discharging code* (Graziotin et al., 2018). So, a chain of consequences can repeat again and again: TD lowers down the developers' morale, and low morale produces low-quality code, which may introduce TD (Olsson et al., 2020). These kinds of findings suggest that it is crucial to understand software practitioners as human beings and how their morale can affect the overall trade-offs between short-term and long-term benefits.

Software development is a socio-technical aspect, and its progress depends on both the social and technical aspects (McLeod & Doolin, 2012). Most software developers work in collective teams with one or multiple supervisors, project leaders, and engineering managers. Morale and team collaboration in the software development process has been found to be highly effective in software development life cycles and are now considered the best way to tackle any project from requirement gathering to deployment, including the refactoring process. A study (Graziotin, Wang, & Abrahamsson, 2014) indicates that happiness and motivational factors are directly related to developers' productivity and software quality. Furthermore, research from the management science domains suggests that there is a stronger correlation between employees' morale and their productivity at work (Ghanbari et al., 2017).

Laribee (Laribee, 2009) in the article, mentioned that the moment when you cannot become productive is excruciating in software engineering, which unknowingly decreases developers' potential productivity. The pain of not being constructive affects developers' state of mind, reduces team morale, and causes productivity to back. So, the study (Laribee, 2009) explains that *"besides the obvious economic downside, there is a real psychological cost of technical debt which ultimately influences developers' productivity."*

(Hardy, 2010) suggests that if any individuals have low morale, their time spent performing their regular tasks is relatively less than the time they spent managing workload. Developers'

morale and productivity can be increased significantly by removing unnecessary reworks (waste, for example, repaying debts). Continuous improvements make developers more proud and motivated, which positively influences their morale, and as a result, higher productivity can be achieved (Hardy, 2010). Previous studies have defined technical debt as a short-term solution (for example, compromising documentation, code quality, testing, or other doubtful budget-saving activities) that lowers developers' morale and productivity in the long run (Hardy, 2010).

2.5 Conceptual framework defining a relationship between TD aspects, developers' morale, and productivity

So far, a few researchers have investigated and concluded that technical debt in software engineering impacts developers' morale, and the morale affected by technical debt influences the developers' overall productivity. However, how specific aspects of technical debt affect developers' morale, happiness, and productivity still need more evidence (Olsson et al., 2020). So, this study seeks to fill out and strengthen the research gap that exists between the technical debt and its' human aspects (developers' morale). Figure 1 below represents the conceptual framework of this study.

The conceptual framework shown in figure 1 below is conceptualized to answer research questions with the help of literature discussed in chapters 2.4.2, 2.4.3, 2.4.4, which have discussed how different aspects related to the occurrence and management of technical debt affects the developers' morale and productivity with the help of previous studies. The occurrence of technical debt has been discussed based on two aspects: introducing TD and criticism for introducing TD. Literature shows that introducing TD has negative influences on developers' morale and productivity. Likewise, not receiving criticism for introducing technical debt could be a source of motivation for developers.

On the other hand, TD management has been discussed based on four different activities: repaying TD, team collaboration for repaying TD, review, and praise for repaying TD. Literature shows that TD management positively affects the developers' morale and productivity. However, literature mentions that specific activities of managing TD, like repaying TD, could be frustrating for developers.

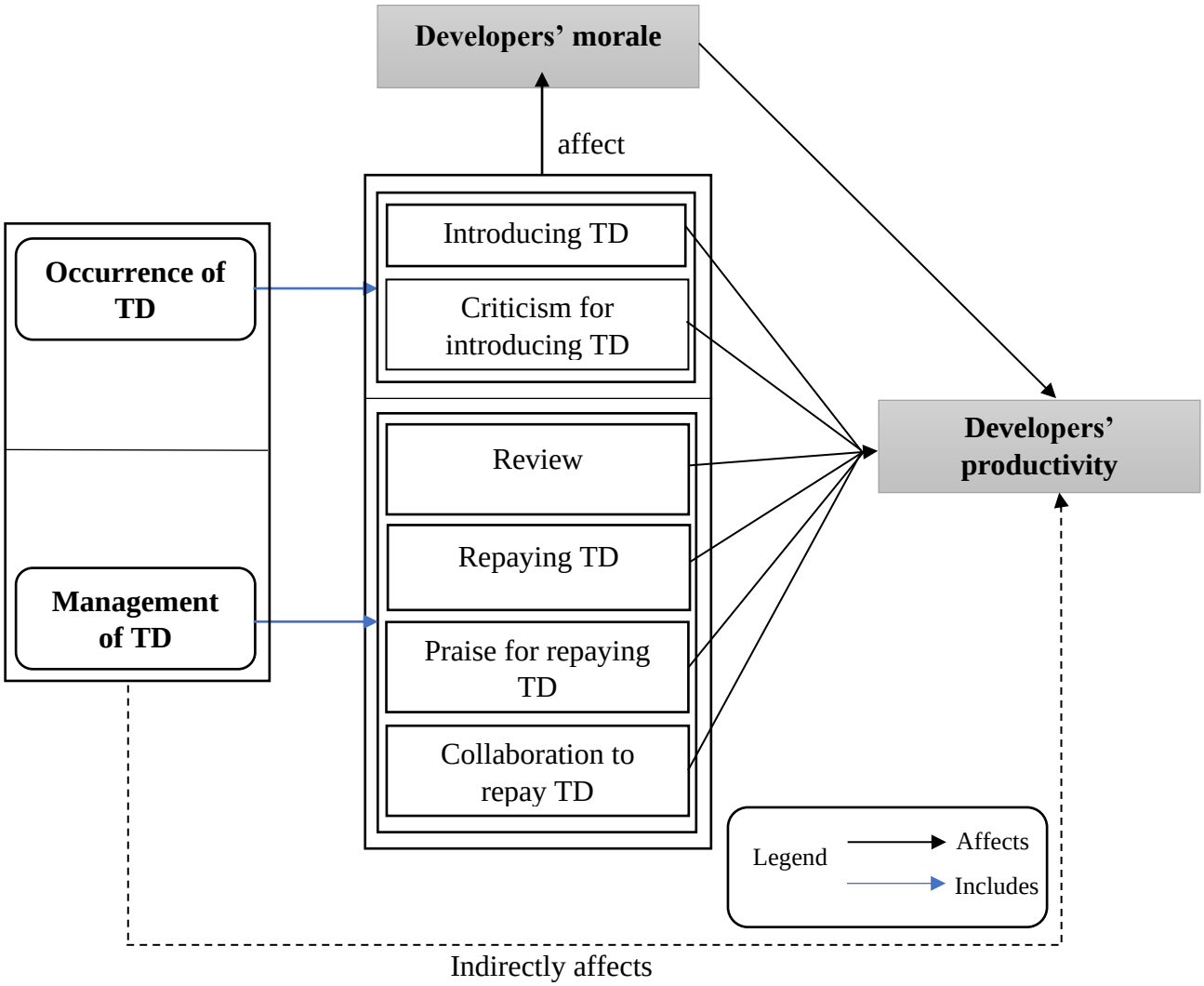


Figure 1: Conceptual framework explaining the relationship between aspects of TD, developers' morale, and productivity.

The above figure 1 summarizes the relationship between TD aspects, developers' morale, and productivity and sets a path for further study to answer the research question. Chapter 4 discuss the research methodologies, data collection, and analysis. Section 5 presents the result, while section 6 discusses the results of this study based on the conceptual framework.

3. Research methodologies

This chapter summarizes the research method used in this thesis, the right reasons for selecting the methods, and the research design's overall structure. First, a preliminary study and the type of research used are presented. Then I will explain the research design for this study, followed by details of data collection, including literature reviews, questionnaires for interviews, and detailed processes about semi-structured interviews (i.e., informant selection and transcription). Lastly, data analysis is presented.

3.1 Research

Collecting information, documenting facts, and searching for information are all terms that are commonly used when it comes to research. The process of gathering, evaluating, and interpreting data to comprehend a phenomenon is known as research. The research process is methodical, which follows a set of steps for establishing the goal, organizing the data, and disseminating the results within the existing frameworks and guidelines (Williams, 2007). (Williams, 2007) likewise mentions that the researchers can use the framework and procedures to find out what to be included in their research, how to conduct it, and what types of interferences are likely based on the data acquired. The three standard research techniques are quantitative, qualitative, and mixed methods which I will discuss later in this chapter.

This research is not so unique and different from other academic researchers regarding the research methods used. However, the outcome of this study can add up a brick in the research area of technical debt, mainly in the impacts on human aspects related to TD. The research highly focuses on the human aspects of technical debts, which are comparatively less spoken in the current research area of technical debts. In order to explore these aspects, I have formulated two main research questions, which I have already mentioned in section 1.3. Afterward, I decided to perform a literature review before doing interviews. I tried to explore as much as I could with prior literature related to human aspects of technical debts. After doing proper literature reviews, I got some ideas on this field and prepared a questionnaire in different phases with the supervision of experts. Lastly, I interviewed 11 IT professionals from various IT companies with different roles from Nepal and Norway.

3.2 Research Design

A research design is a strategic framework for action that connects the research objective to the research's execution or implementation (Durrheim, 2006). Durrheim explains a research design as a strategy that directs the arrangement of conditions for data collection and analysis in a way that tries to combine relevance to the research goals with efficiency in the procedures. A research design should demonstrate a plan that describes how the research will be carried out in such a way that it responds to the research question. Figure 2 shows the different stages of a research process, demonstrating the role of research design as a bridge between research objectives and execution of research (Durrheim, 2006).

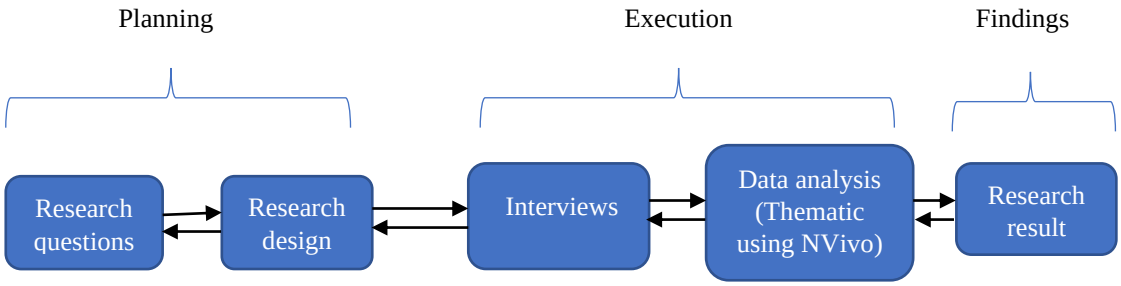


Figure 2: Research design (Durrheim, 2006)

For this study, I first conducted preliminary research in phase 1, where I performed research on prior literature, research, or studies. I found technical debt is a highly growing research area, and researchers are mainly paying attention to the economic, organizational, and technical aspects of TD. However, there is a research gap between technical debt and human aspects because very few studies have researched the impacts of TD on human aspects of technical debt, such as morale. So, throughout the studies, I came to know some important questions (such as: How do developers feel when they have introduced TD? how do they feel when they have to refactor those debts later? How does TD play a role in their motivation? How does a change of morale after introducing TD impact a developers' productivity?) still need more investigation. So, this research is highly dedicated to investigating how TD influences developers' morale and productivity.

After finding the research objective, in phase 2, I collected data through in-depth semi-structured interviews. Questions for the interviews have been prepared based on those potential research opportunities found in phase 1 with the help of continuous discussion and supervision of supervisors. Once the interviews had been completed, all of them were transcribed in the text and then coded into different categories. Then these categories were

mapped into themes as the impact of TD on developers' morale and productivity. Data analysis was done through thematic analysis with the help of NVivo 12 Pro.

A detailed explanation of phase 1 (i.e., preliminary study) is presented in chapter 3.3; Chapter 3.6 explains the data collection process. Chapter 3.7 describes the data analysis process with the sample of data coding and mapping into the themes.

The research design illustrated in figure 3 below presents the overall structure of the research process used in this study. Gray boxes represent different phases of data collection, while the white box represents the data analysis process. Likewise, rectangle boxes are methods used to collect data, ellipses represent the result, and arrows represent the information flow from different activities to the result.

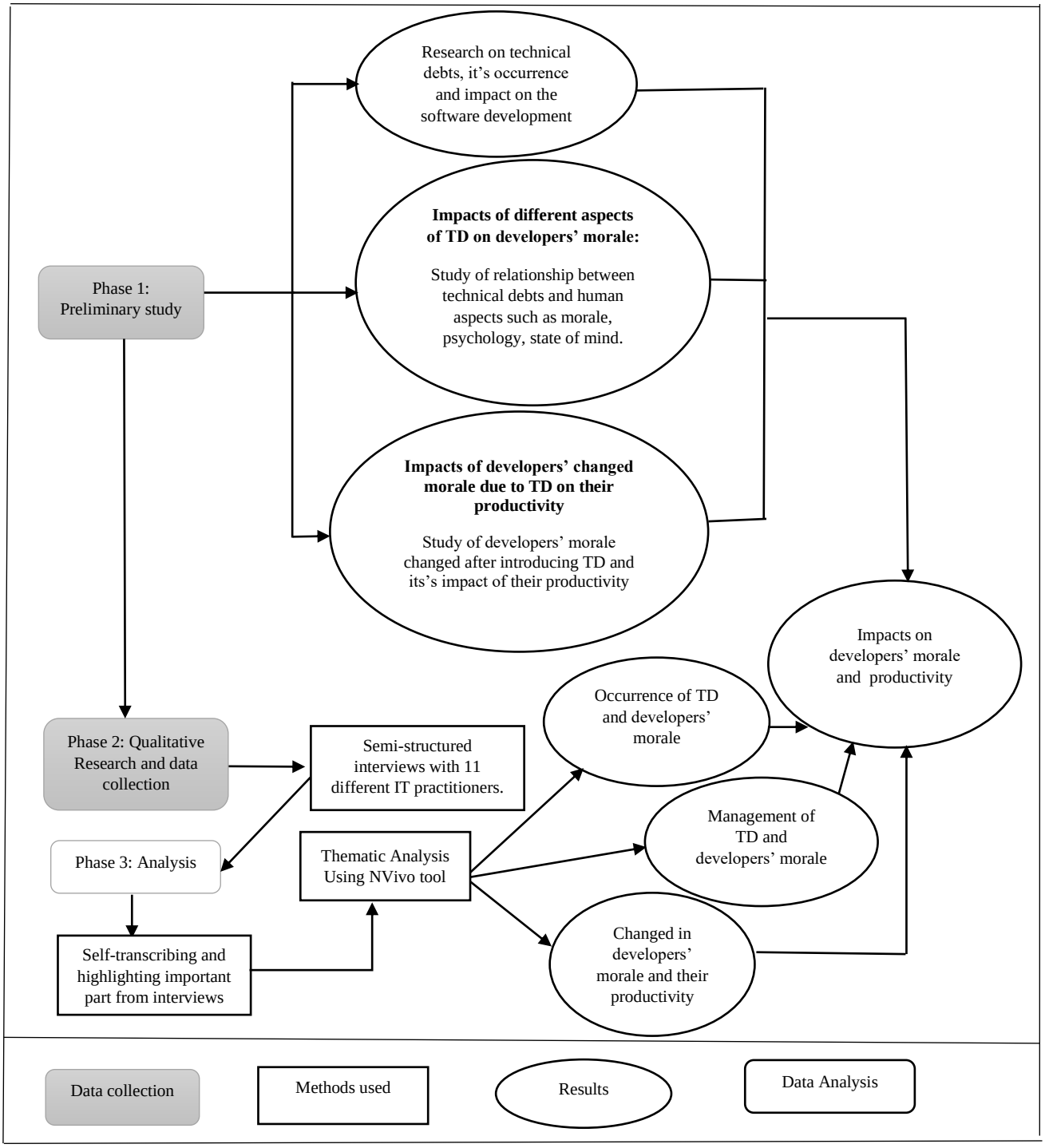


Figure 3: Research design for this study

3.3 Preliminary Studies

I have already mentioned that there are a few potential opportunities to research related to the impact of technical debt on developers' morale and productivity. However, I was familiar with the concept of technical debts in a lecture on IN5140 - Smart processes and agile methods in software engineering, which I have studied in my first semester. Later, I started the journey of writing the thesis from literature reviews.

Literature reviews

A literature review is an overview of previously published works on a particular subject. Conducting a literature review is crucial for developing a research proposal, combining existing information about a specific subject, identifying knowledge gaps, and assessing how your research may add to further understanding (Winchester & Salji, 2016).

After conducting a review of the literature, I realized that the impact of technical debt on developer morale and productivity could be studied in terms of various aspects of technical debt, including introducing TD, criticism for introducing TD, the review process, repaying technical debt, praise, and team collaboration. This is why the study concentrated on these technical debt-related aspects and their impact on developer morale and productivity.

This study mainly uses the following databases to search literature

Table 1: Database used to search literature

Database	**Website**
Google Scholar	https://scholar.google.com/
ScienceDirect	https://www.sciencedirect.com/
IEEE Xplore	https://ieeexplore.ieee.org/
ACM	https://dl.acm.org/
SpringerLink	https://link.springer.com/
Oria	https://oria.no/

While searching the database to research the related article, I used the following keywords and terms.

Table 2: Sample of keywords used to search literature

Technical debt, aspects of technical debt, human aspects related to technical debt, risk of technical debt, morale, feeling, emotions, psychology, motivation, self-worth, performance, productivity, risk, quality, criticism, praise, developers' morale, risk of TD, quality comprises because of TD, the impact of technical debt on developers' morale, developers' productivity, the impact of TD on developers' productivity, the impact of morale on productivity, team collaboration, team collaboration to manage TD, technical debt management

The intensive literature search resulted in the publication of numerous articles. Afterward, I initiated the process of identifying the most pertinent articles. I conducted snowball sampling to identify the most relevant articles.

Snowball sampling method

Snowballing is a term that refers to the process of identifying more articles using a literature reference list or citations. On the other side, snowballing examines where papers are actually referred to and cited. Snowballing as a search technique may be preferable to database searches. Using the references and the citations respectively is referred to as backward and forward snowballing (Wohlin, 2014). Thus, snowballing sampling makes my task more manageable and effective in sorting out the relevant articles. Once I found a relevant article in the database, I always looked at the article's reference list, which helped me discover a plethora of additional helpful articles. Additionally, when I searched and found a relevant study in a database, I also browsed the recommended articles by databases. This is how I identified appropriate studies for my thesis.

The following table 3 contains a sample list of literature studied along with their sources and brief descriptions. In the table, I mentioned only 12 studies. However, this thesis uses plenty of articles, research papers, and other literature relating to the human aspects of technical debt, which are not mentioned in the table but listed in the reference list.

Table 3: Sample list of used articles

S.N	Literature	Source	Description
1	The WyCash Portfolio Management System	ACM SIGPLAN OOPS Messenger , ACM dl.acm.org	A case study on redesigning architectural decisions to speed up software development
2	An exploration of technical debt	Journal of Systems and Software, ScienceDirect	The study explores the detailed concept of technical debt using MLR and interviews
3.	Looking for a piece of mind	2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement	Exploratory field research to determine the extent to which TD and its management may affect the morale of software developers
4	The influence of Technical Debt on software developer morale	The Journal of Systems and Software, ScienceDirect	Exploratory study to understand how software developers' morale is influenced by TD present in the software. Data was collected through different rounds of interviews and surveys with IT practitioners.
5	A Balancing Act: What Software Practitioners Have to Say about Technical Debt	IEEE Software, IEEE Xplore	An exploratory field study based on interviews with 35 software developers, explains how practitioners define technical debt

S.N	Literature	Source	Description
6	The Sources and Approaches to Management of Technical Debt: A Case Study of Two Product Lines in a Middle-Size Finnish Software Company	SpringerLink, International Conference on Product-Focused Software Process Improvement	A case study of a medium-sized Finnish software company to comprehend the causes and consequences of technical debt
7	Technical Debt: The Ultimate Antipattern - The Biggest Costs May Be hidden, Widespread, and Long Term	IEEE Xplore, Conference: 2014 Sixth International Workshop on Managing Technical Debt	A study that examines the hidden cost of taking shortcuts to developers' morale and productivity.
8	Measuring affective states from technical debt: A psychoempirical software engineering experiment	arXiv:2009.10660	A psycho empirical software engineering experiment investigating the relationship between technical debt and affective states (feelings, emotions, and moods) of software practitioners.
9	A systematic mapping study on technical debt and its management	Journal of Systems and Software, ScienceDirect	A systematic mapping study on TD and TD management (TDM) and making a classification and thematic analysis on these studies.
10	What happens when software developers are (un)happy	Journal of Systems and Software, ScienceDirect	A study that examines what happens when developers are happy and unhappy while developing software.

S.N	Literature	Source	Description
11	Towards Effective Technical Debt Decision Making in Software Startups: Early-Stage	ACM SIGSOFT Software Engineering Notes, ACM	Multiple retrospective case studies in different software startups willing to explore the relationship between TD decisions and the success or failure of software startups.
12	Investigating Technical Debt Folklore Shedding Some Light on Technical Debt Opinion	2013 4th International Workshop on Managing Technical Debt (MTD), Conference, IEEE Xplore	A survey collecting and analyzing many assertions made by practitioners about technical debt on online pages, blogs, and published papers.

3.4 Case Study

The case study is the most widely used method in academia for researchers interested in qualitative research (Baskarada, 2014). A qualitative case study is a research methodology that aids in exploring a phenomenon within a specific context using numerous data sources. It does so by using a range of lenses to show multiple sides of the phenomenon. In a case study, a real-time phenomenon is investigated in its natural setting with the understanding that context makes a difference (Rashid, Rashid, Warraich, Sabir, & Waseem, 2019). Gerring argued and defined a case study as *"an intensive study of a single unit with an aim to generalize across a larger set of units"* (Gerring, 2004). A case study is an empirical technique. This refers to a method for setting research questions, collecting data, analyzing the data, and presenting well-defined and scientific conclusions (Perry, Sim, & Easterbrook, 2006).

A study (Runeson & Höst, 2009) presents five significant steps to be walked through while conducting a case study.

- *"Case study design: objectives are defined, and the case study is planned.*
- *Preparation for data collection: procedures and protocols for data collection are defined.*
- *Collecting evidence: execution with data collection on the studied case.*
- *Analysis of collected data*
- *Reporting"*

The research goal of this thesis is all about studying developers' morale, psychology, natural behavior, and reaction. For this, we need to extract real-time, more detailed, and more profound descriptions, opinions, feelings, and perspectives from the developers. As the case studies are by definition conducted in real-world settings and thus have a high degree of realism, and primarily based on the qualitative data, I believe this could be a suitable method for my research.

The qualitative interview is one of the most used and appropriate data collection methods in case studies (Rashid et al., 2019). So, in this research, I performed a case study (which is defined in detail in chapter 4) by conducting semi-structured interviews with eleven IT professionals from different IT companies to investigate the impact on their morale and productivity due to technical debt. I tried my best to include the broader domain of opinions

and perspectives from developers on technical debts and its' human aspects, which helped me to extract a better understanding and outcome to answer the research questions.

I followed all the major steps of case studies. Firstly, the objectives (represented by two research questions) of this research have were defined. Secondly, I conducted interviews to collect the data. Then I tried to answer the research question by making different inferences with the help of collected data and prior literature. After that, I performed a thematic analysis with the use of software to analyze the data. Lastly, I extracted the result from the analyzed data. So, in this way case study was performed.

3.5 Qualitative Research

Research is the process of collecting data or information, analyzing and interpreting those data to get the required results. The three most common methods used in the research are quantitative, qualitative, and mixed methods. The researchers select the methods based on the type of data they need to answer their research questions. Researchers usually prefer quantitative research methods if they need numerical data to answer the question. If textual or subjective data such as elaboration, explanations, opinions, perspectives are required, researchers can select a qualitative research approach. On the other hand, mixed methods research is used when researchers need both numerical and textual data (Williams, 2007).

Qualitative research may also be defined as an unfolding model that takes place in a natural setting and allows the researcher to build a level of detail through active participation in the actual events (John W Creswell, 1994). Qualitative research entails the utilization of collected data to describe, explain, and interpret those data. The description in qualitative research is less structured because it formulates and establishes new theories or hypotheses (Williams, 2007). Qualitative research is also known as an effective approach that takes place in a natural setting and allows the researcher to develop a level of detail by being deeply involved in the actual experiences (Williams, 2007).

Qualitative research is a method of investigating and comprehending the meaning that individuals or groups attach to a social or human issue. Emerging questions and processes are part of the research process, as are data gathered in the participant's environment, data analysis that builds inductively from specifics to broad themes, and the researchers' interpretations of the meaning of the data (John W. Creswell, 2009).

This thesis aims to find out how developers' morale changes or is being affected in a natural way due to the occurrence of technical debt. In a broad sense, this study aims to determine the developers' behavior in a natural context. Hence, the qualitative research method seems to be more appropriate based on the research problems and my own personal experience. The qualitative research in this study is implemented by conducting 11 interviews with 11 IT professionals from different software companies in Norway and Nepal.

3.6 Data Collection

3.6.1 Interviews

In case studies, data collecting through interviews is critical. In interview-based data collection, the researcher asks interviewees a series of questions about the case study's topics of Interest (Runeson & Höst, 2009). The qualitative interview is one of the most common and essential data gathering tools in qualitative research. It is used in case studies, in action research, and in grounded theory. In qualitative interviews, the researcher performs face-to-face interviews with participants, conducts telephone interviews with participants, or conducts focus group interviews with six to eight interviewees in each group. These interviews consist of a series of unstructured, often open-ended questions, aiming to elicit the participants' thoughts and ideas (John W. Creswell, 2009). There are different types of interviews, but these can be mainly divided into three categories based on the degree of structuring (Longhurst, 2003).

Unstructured Interviews: The interview questions in an unstructured interview are created as general concerns and interests from the researcher. The interview dialogue will evolve based on the subject's and researcher's mutual interests in this situation. There are no pre-defined questions (Runeson & Höst, 2009).

Semi-structured Interviews: A semi-structured interview has pre-planned questions, but they are not mandatorily asked in the same order as they are listed. These interviews allow both researchers and interviewees to improvise, explore, and bring up new ideas based on the conversation. Semi-structured interviews are most common in case studies. (Longhurst, 2003), (Runeson & Höst, 2009).

Structured interviews: In a structured interview, all questions are planned in advance and strictly regulated according to their order and guidelines. It follows a predetermined and standardized list of questions, and there is no room for improvisation. In many ways, a fully

structured interview is similar to a questionnaire-based survey (Longhurst, 2003; Runeson & Höst, 2009).

Most of the previous research (Longhurst, 2003; Runeson & Höst, 2009) recommends that semi-structured interviews are the most used and appropriate method for collecting data. So, I decided to go for semi-structured interviews. As mentioned above, the semi-structured interviews are very open and allow both interviewer and respondent to improvise and explore new ideas while talking. I wanted my interviewees to speak freely about their opinions, experiences, perspectives, and feelings on the impact of technical debts on their morale and productivity. In addition, semi-structured interviews make it easy to explore the broader aspects by allowing improvisation upon required. I did not want to resist them in a boundary of a set of questions by performing a fully structured interview with predefined questions.

On the other hand, it will not be easy to cover all the aspects that I was looking for by conducting an unstructured interview. I have not had that much experience with the interview-taking process so that I might forget some crucial aspects without a guideline. Hence, I decided not to go with unstructured interviews.

The year 2021 is fully affected by the COVID-19 pandemic, so it wasn't easy and possible to take interviews by meeting in person. As an option, I used 'zoom' as a communication means to conduct the interviews. It is not practicable to note down all the answers from the interviewee during the interviews process, so I recorded interviews in audiovisual format.

3.6.2 Informant Sections

Data in qualitative research are fundamental, so selecting good informants for collecting reliable data is not an easy task to do. As a result of their personal skills and experiences, key informants can provide more information and a deeper insight into what researchers are looking for (Marshall, 1996a). Since it is rarely practicable, effective, or ethical to examine entire populations, selecting a study sample is crucial in any research (Marshall, 1996b).

In research, sampling is the process of selecting a group of people from a larger population for a research purpose (Heurich & Vignali, 2015). The finding from the selected sample can be generalized to the whole population. One of the most common approaches for sampling is the random or probabilistic sampling method which is mainly used in quantitative studies (Marshall, 1996b). In a probabilistic sampling method, the nature of the population is defined, and all members have an equal chance of selection. Stratified random sampling and area sampling are variants of random sampling, which allow subgroups to be studied in

greater detail (Marshall, 1996b). According to Marshall (Marshall, 1996b), probabilistic sampling is inappropriate for qualitative studies. Some informants can be richer than others, and those with richer information are more likely to provide better insight and understanding for the researcher (Marshall, 1996b). In contrast, "*choosing someone at random to answer a qualitative question would be analogous to randomly asking a passer-by how to repair a broken-down car, rather than requesting a garage mechanic*" (Marshall, 1996b, p. 523).

Judgmental, also known as purposive sampling, is the most common sampling method used in qualitative studies. In this process, researchers identify the most appropriate or productive informants to answer the interview and research questions. On the other hand, the convenience sampling technique selects the most accessible informants, which might be easier to perform and least costly. This, however, may lack intellectual legitimacy and result in less qualitative data (Marshall, 1996b).

After conducting a thorough literature study and being familiar with the various sampling methods, I chose a non-probabilistic sampling strategy in which informants are selected using purposive sampling methods. The main criteria for selecting informants were as follows:

- The informant should be currently working in an IT/software-related company.
- The informant should have a basic concept about technical debt.
- The informant should have at least two years of working experience in the field of software engineering.
- The informant should be open to sharing their natural feeling/opinions/perspectives related to the occurrence of TD.

Based on these criteria, I contacted interviewees by sending them emails or calling them. Altogether eleven interviews were conducted from 11 different companies of different roles. These informants either work in Norway or Nepal. The general information (excluding sensitive and personal details) about informants and their companies are described in chapter 4.

3.6.3 Questionnaire

A questionnaire should be based on the objective of the research study, i.e., with formulated research questions. Questions can be open or closed. Open questions allow respondents to present their broader range of answers, while close questions offer a limited set of alternative answers (Runeson & Höst, 2009).

After a proper literature review, I found a research gap between technical debt and its' human aspect. Then, to address such shortcomings, I devised two research questions. For this qualitative study, I need to collect the data by conducting semi-structured interviews. Given that my research concerns developers' morale, I prepared the initial draft of my questionnaire based on the three different antecedents of morale (affective, future/goal, and interpersonal) which are well described in section 2.6.1. After preparing the first draft, the questions were tested with experts.

(Myers & Newman, 2007) recommends the script in a semi-structured interview should involve, at a minimum:

- "Preparing the opening – introducing yourself.
- Preparing the introduction – explaining the purpose of the interview.
- Preparing the key questions.
- Preparing the close – if needed, asking permission to follow-up, or asking who else the interviewee recommends might be interviewed."

I prepared my questionnaire by following guidelines given by Myers and Neman (Myers & Newman, 2007), which is described briefly as follows:

- **Opening:** I open the interview with my introduction, including my name, what, and where I am studying.
- **Introduction:** I briefly described the research topic and explained the purpose of the study. Since I took the interviews in the zoom, I read a consent form on their behalf, and they should agree on it before conducting the interview.
- **Startup questions:** I had also prepared some general questions to know them better, such as their experiences, current role, educations. For example:
    - What is your highest education level?
    - What is your role at this company?
    - How many years of experience do you have in this role?
- **Key questions:** There were altogether 13 key questions in my questionnaire. These questions are related to quality, the occurrence of technical debt, developers, morale, productivity, and collaboration. These questions aim to cover my research questions. Some key questions are as follows;
    - How do you feel when someone reviews your work?
    - What are the positive and negative feelings of introducing technical debts?

- How does the change of morale that happens after introducing TD influence your performance?
- In the future, you might need to pay the debts that you have accumulated in the past. How do you feel when you actually spend time repaying debts instead of developing or implementing new features?
- How do you feel when your colleagues help you to fix the debts?

- **Closing:** At the end of the interview, I always used to ask for any suggestions or recommendations that I could add up to make this research more relevant.

3.6.4 Interview process

I started conducting interviews from the end of May 2021 and took almost 20-25 days to complete all 11 interviews. So, I can say I collected all the data over a month. As we are all aware of the Covid19 pandemic, I could not travel to meet interviewees, so I interviewed them through an online platform called zoom[1]. It was a pretty different experience taking interviews using online platforms. However, we were very used to it due to this pandemic, so I did not experience any difficulties. Regardless of whether these interviews were held online, I always attempted to retain a professional demeanor during the interview process.

I began the interviews by introducing myself, stating the research goal, and reading the consent form. Once they accepted the consent, I started with general questions (startup questions), including their general information like education level, the field of study, current role at the job, and experience. After that, key questions used to be discussed, which were more open-ended and focused on technical debt, morale, and productivity. At last, I ended the interviews with a few closing questions. Since interviews were semi-structured, the questionnaire was critical in ensuring I did not overlook any pertinent questions to ask respondents. On average, an interview lasted for 20 to 30 minutes. The shortest interview has lasted 18 minutes, while the longest was for 32 minutes.

Because interviews were quite lengthy, it was challenging to note all participants' responses throughout the interview process. As a result, with the consent of respondents, I recorded interviews in an audiovisual format to ensure that no crucial information was missed.

[1] Zoom is a cloud-based peer-to-peer software platform and is used for teleconferencing, telecommuting, distance education, and social relations.

3.6.5 Interview Transcriptions

Transcribing those extended interviews into text is a time-consuming and frustrating process in itself, so most researchers usually do not like this process. Transcription is an essential process to get the qualitative data for the researchers conducting advanced data analysis (Matheson, 2007). Since semi-structured interviews often produce a tremendous amount of text, many researchers experience transcribing process as a tiresome, lengthy, and challenging process that needs specialized skills, patience, and physical ability (Matheson, 2007).

The next step in my research after conducting interviews is transcribing those interviews. I tried to transcribe interviews immediately after finishing them, which helped me not to forget the essential information that respondents mentioned. I transcribed all the interviews on my own. I listened to those recorded interviews many times and tried to document them in a word file. The proper transcription of interviews is really important because inappropriate or inadequate transcriptions from audio or digital recordings can negatively affect the analysis process(McLellan, MacQueen, & Neidig, 2003). As there is no universal format for transcribing interviews (McLellan et al., 2003), I decided not to include pauses, laughter, and voices like “hmm,” “huh,” and “umm” while transcribing.

3.7 Data Analysis

Data analysis is a process of collecting data and developing a conclusion, insights, or interpretation that addresses the research questions or supports the decision-making process. The process should be iterative, starting by developing a basic sense or understanding of the research context and perspectives or opinions from the people being studied for the research. Then it should be refined with more data in the following iterative (Aydin & Anderson, 2005).

Qualitative data analysis methods are often utilized in case study research since it is a versatile research method. The analysis' main goal is to draw conclusions from the data while maintaining a transparent chain of evidence. A reader should be able to follow the derivation of results and findings from the collected data, which is referred to as the chain of evidence(Runeson & Höst, 2009).

I have collected the data from the interviews, so I did one of the qualitative data analysis methods called thematic data analysis using NVivo. *"Thematic analysis (TA) is a method for identifying, analyzing, and interpreting patterns of meaning ('themes') within qualitative data"* (Victoria & Virginia, 2017). As thematic analysis has been chosen as a data analysis method for this study, firstly, I transcribed the recorded interviews (qualitative data) into text then openly coded those texts by highlighting essential and meaningful parts of the interviewee's answer. After that, all the transcribed pdf files, including openly coded data, were uploaded into NVivo 12 pro software [2] (a famous qualitative data analysis software). In the thematic analysis, coding is an important step. Codes can be defined as the building block for the research theme. They are the finest units of the analysis process that reflect the information or answers relevant to research questions (Victoria & Virginia, 2017). Since I read those transcriptions of the interviews 4/5 times, it helped me sort out the codes and subcodes from the interview's transcription using NVivo. Those codes and subcodes are then classified into categories that ultimately determine the themes for this research study.

For example, an interviewee's answer, *"I will have my inner drive that it would give me some guilty feelings in this regard because it is like I am actually cheating." is coded and categorized as* 'Introducing TD' because the answer reflects the developers inner feeling after introducing technical debt. This answer is finally mapped into one of the themes: "Aspects of TD affecting developers' morale." Likewise, another answer, "*It depends on the situation, if*

[2] NVivo is a qualitative data analysis software developed by QSR International, www.qsrinternational.com

introducing TDs helps your work then the morale will be high, and the high morale and good mind will increase your performance." is coded into another category from the second theme, "Introducing TD." This is coded and classified in that category because the interviewee's answer relates to how the change of morale after introducing TD affects the developers' performance. Since this code is related to productivity, it is mapped into the "aspects of TD affecting the developer morale that also affects developers' productivity" theme.

Below, figure 4 illustrates how the interviews were transcribed into different codes, subcodes, categories, and themes. The interview's answers are mapped into two main themes through subcodes and codes. These themes cover the research questions of this thesis with the help of codes, subcodes, and interview quotes.

For the first theme, codes were categorized into five categories: 'reviewing process (includes code review),' 'Introducing technical debt,' 'Getting criticism or praise,' 'Repaying technical debt,' and 'team collaboration on fixing TD.' These codes are then mapped into the theme: aspects of TD affecting developers' morale. The categorization of the codes has been done based on the research aim, prior literature, my personal experience, and interviews as they recommend that these five categories are directly related (or affect) to developers' morale.

On the other hand, the same five categories of codes are mapped into the second theme: "Aspects of TD affecting the developer morale that also affects developers' productivity." These codes are the same for both themes because all of them affect developers' morale and productivity. For example, getting criticism or praise first affects the developers' morale, which then affects their productivity. Likewise, Repaying technical debt affects morale and productivity at the same time.

It might not be effective in generalizing the interview quotes directly into higher categories (i.e., codes). So, every interview quote was categorized into more specific subcodes before categorizing them into codes. Those subcodes will help me to analyze the data and write the result more efficiently. Here in figure 4 below, each category of codes has one subcode and one interview quote. It is unlikely to show all the interview quotes and subcodes since there are too many. For example, the code " Introducing technical debt" has 16 subcodes and 29 interview quotes. So, figure 4 just represents samples of the subcodes and interview quotes.

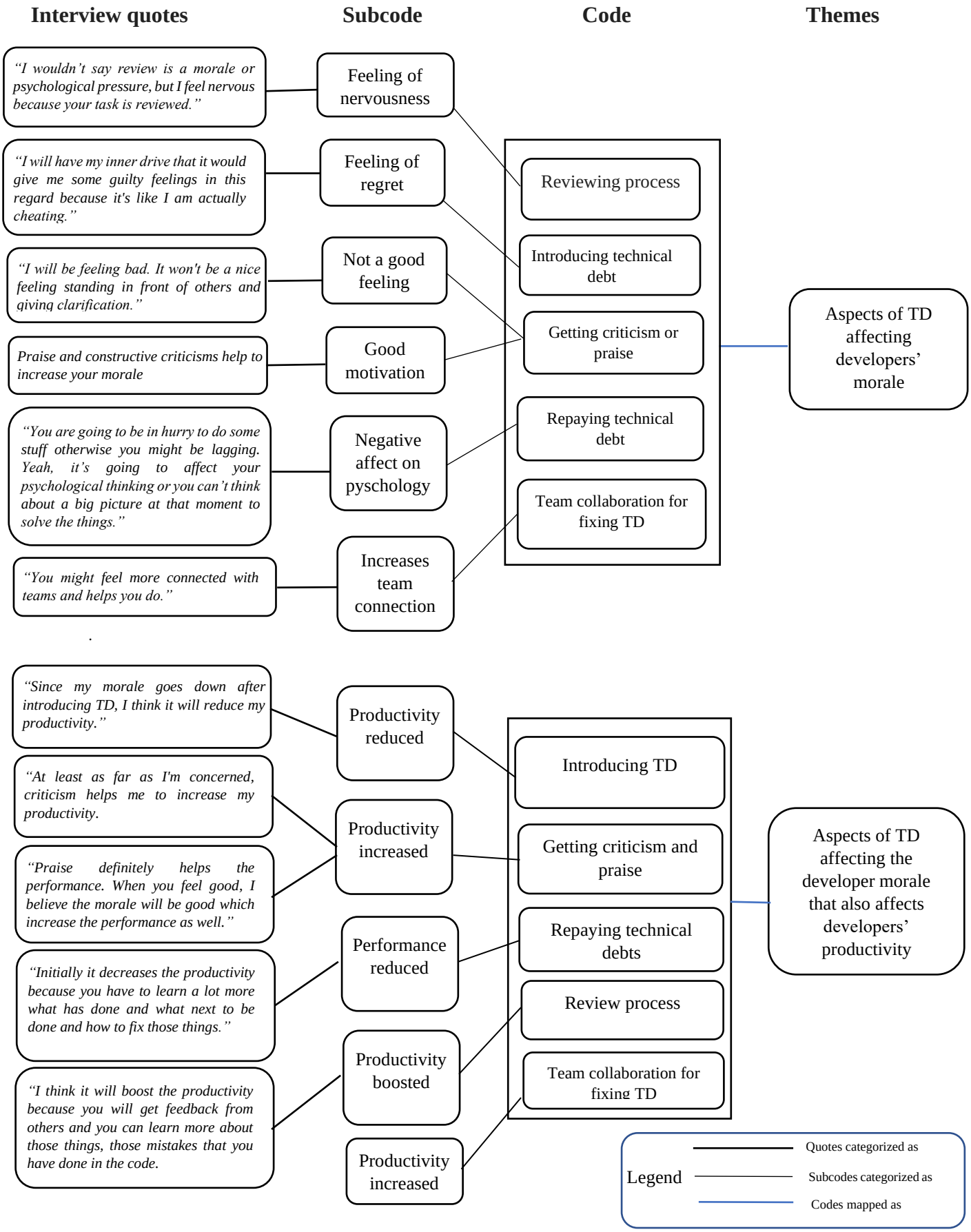


Figure 4: Sample of data coding and mapping into the themes

It is unlikely to show all the codes and subcodes under those nine categories in Figure 4 above. So, figure 5 below shows the sample of codes and subcodes used to map interview quotes. However, appendix B will provide more codes and subcodes used to analyze interview data in NVivo.

Name	Files	References
Getting criticism or praise (Morale)	11	34
Getting criticism or praise (productivity)	7	9
Introducing technical debt	10	29
An open feeling	1	1
Feel demotivated	2	2
Feel insecure	1	1
Feel satisfied	1	1
feel unsatisfied	3	3
feeling of nervousness	1	1
Feeling of regret	2	2
Feels relieved	1	1
Guilty feeling	1	1
Have mixed feelings	2	2
Morale will be high if TD works	1	1
Not a bad feeling	1	2
Not a good feeling	4	5
Not satisfied about the way	1	1
Positive feeling on completing task	3	3
Positive if it works well	1	2
Morale after introducing TD (Productivity)	10	21
Repaying technical debt (morale)	11	38
Affect psychological thinking	1	1
Fixing TD	10	14
Good feeling	1	1

Figure 5: Sample of thematic analysis performed to analyze interview data using NVivo

4. Description of the studied interviewees

In this section, I will describe the details of the interviewees who were interviewed during the study. I will not reveal their personal identity anymore but will present their general information.

Respondent I: Respondent 1 is a software developer with more than two years of experience in the current role. However, he has more than six years of experience in the field of software engineering. He has completed his master's degree specializing in computer science, and he is currently working in a software company based in Norway.

Respondent II: Respondent 2 has more than four years of experience as a chief technology officer (CTO) and team lead which is his current role. However, he has overall six years of experience in the software industry. He has completed his bachelor's in software engineering. Currently, he works in a Nepal-based IT company.

Respondent III: Respondent 3 is a software engineer who has a bachelor's degree in software engineering. He is currently working as a project manager for internal products for five years. He has more than seven years of experience in the field of the software industry. He is working in a Nepal-based IT company.

Respondent IV: Respondent 4 is a project manager with a minimum of five years of experience in his current role. However, he has seven years of experience in the IT field. He has completed his bachelor's degree in software engineering. He is currently working in a Nepal-based IT company.

Respondent V: Respondent 5 has been working in a Norway-based software company as a front-end developer for five years. However, he has seven years of overall experience in software engineering. He has completed his master's degree in Information technology.

Respondent VI: Respondent 6 is a senior project manager who has been recently promoted from the manager. So, he has only seven months of experience in this role; however, he has more than nine years of experience in the field of software engineering. He is a bachelor's degree graduate in software engineering and currently working in a Nepal-based software company.

Respondent VII: Respondent seven is a bachelor's degree graduate in computer science. He is currently working as a full-stack developer in a software company based in Norway for

two years. However, he has overall five years of experience in the field of software engineering.

Respondent VIII: Respondent eight is an android developer with more than five and half years of experience in this current role. However, he has been working in the field of IT for seven years. He has done his master's degree in computer science (ICT design). Right now, he is working in a Norwegian IT company.

Respondent IX: Respondent nine has almost five years of experience in his current system developer role, and he has been working in the IT industry for five years. He has earned his master's degree in computer science. Right now, he is working in a Norwegian company as a system developer.

Respondent X: Respondent ten is a mechanical engineer; however, he is a senior software developer right now. He has been promoted recently, so he does have less than a year of experience in this role. But then, he has more than twenty years of overall experience in the field of the IT industry. He is working in a Norway-based IT company.

Respondent XI: Respondent eleven is the last participant, a senior infrastructure sub-engineer, and has three years of experience in this role. He has overall sixteen years of experience in the field of information technology. He has a master's degree in computer science. He is currently working in a Norwegian IT company.

So, these are the general information about my interviewees that I could share here. Their privacy and confidentiality are strictly maintained. Table 4 below summarizes the details about interviewees who participated in this study. The table presents companies coded using the English alphabet from A to K, their current role, their highest level of education, experiences in the current role, the overall experience in software engineering, and the country where the company is based.

Table 4: Description of the studied interviewees

S. N	Company	Role	Education	Experience in the current role	Overall experience in the Software Engineering	Country
1	**A**	**Developer**	Master	2	6	Norway
2	**B**	**CTO and Product Lead**	Bachelor	4	6	Nepal
3	**C**	**Project Manager for Internal products**	Bachelor	5	7	Nepal
4	**D**	**Project Manager**	Bachelor	5	7	Nepal
5	**E**	**Front end developer**	Master	5	7	Norway
6	**F**	**Senior Project Manager**	Bachelor	1 year (promoted last year)	9	Nepal
7	**G**	**Full stack developer**	Bachelor	2	5	Norway
8	**H**	**Android developer**	Master	5.5	7	Norway
9	**I**	**System Developer**	Master	5	7	Norway
10	**J**	**Senior Software Developers**	Master	Less than a year	20	Norway
11	**K**	**Senior Infrastructure sub engineer**	Master	3	16	Norway

5. Results

The results section should be a widely informative and important piece in any academic or scientific research. This section accurately describes and displays the output from the data analysis. The results section should provide adequate information for others to assess the study's merits and conclusions and laying out a blueprint for future replication (Hahn Fox & Jennings, 2014).

This chapter will illustrate what I found through data analysis of collected data in a structured way. The results are based on the answers from the interviewees to the proposed research questions. Firstly, the impact of the occurrence and management of technical debts on developers' morale from different aspects of technical debt will be discussed, followed by the impact of developers' morale on their productivity via the occurrence and management of technical debt. Ultimately, the relationship between technical debt, developers' morale, and productivity will be summarized in a table.

5.1 The impact of the occurrence of TD on developers' morale and productivity

In the process of finding the impact of occurrence and management of TD on morale and productivity, firstly, I have sorted out the different aspects of TD related to the occurrence of TD that affect the developer's morale and productivity. So, section 5.1 will discuss how various TD aspects associated with the occurrence of TD impacts developers' morale. This chapter also explains how developers' morale affects the developers' productivity via the occurrence of TD.

5.1.1 The impacts of introducing technical debts on developers' morale

The majority of previous research (Ghanbari et al., 2017), (Besker et al., 2020), (Allman, 2012), (Li et al., 2015), (Tom et al., 2013) agrees that the accumulation of technical debt in the software development process is inevitable. So, based on this statement, I asked interviewees a few questions related to the introduction of technical debt and its influence on their morale. Almost all interviewees admitted that they sometimes introduce technical debt, but it depends on the circumstances.

In the process of studying the developers' feelings after introducing TD, some keywords (subcode) like 'demotivated,' 'insecure,' 'unsatisfied,' 'nervous,' 'regret,' 'bad feeling,' 'guilty,' 'relieved,' 'good feeling,' 'satisfied,' 'positive' or 'mixed feeling' were noted down.

All of these keywords reflect the developers' feelings after introducing technical debt. Majorities of interviewees expressed their negative feeling about introducing technical debt. For example, one of the interviewees mentioned that "*So, the first thing is I have the inner drive that would give me some very guilty feelings in this regard because it is like I would be cheating, you know, I am being paid to do my best job. So, if I am knowingly not doing it, I am actually cheating.*" A few of them feel demotivated once they realize their work is going to be dismantled; whatever they previously worked on will be unaddressed, or they need to rework on everything. Some interviewees also feel nervous and insecure because they know the way they did the task might not perform at its optimal.

Usually, introducing TD negatively impacts developers' morale and demotivates them.

Most of the interviewees do not feel good if they introduce technical debt. For example, one of the interviewees said, *"violating the rules might be faster, but the quality standards and rules should be followed regarding the quality of work. I usually violate the quality standards when I am in a rush, but that does not give me a vibe of good feelings."* From the interviews data, we can say that developers were all aware that they were introducing technical debt, which might create a problem or need to rework later. So, they do not feel good in the long run at all by violating quality standards, company guidelines, or other rules.

However, few developers feel relieved, good, or positive at least on completing the task because sometimes they might have pressure to finish their work within the predetermined deadline, cost, or resources. For example, one of the interviewees mentioned that *"If you are targeting to release the product and it has a huge amount of competition in the market, then I do not feel that much worse even I violate the quality standards for a short term."* So, if technical debt helps developers to complete their tasks before the deadline, they might not feel bad immediately. In this regard, an interviewee said, *"Talking about the feelings after introducing TD is like if it works whatever we try to do, then we feel happy and accomplished."*

Technical debts can provide short term happiness or satisfaction for few developers.

Nevertheless, in the long run, when TD backfires, the developers feel like they should not have done it. Thus, few developers took the introduction of TD as a subjective feeling. If it works whatever they are attempting to do, they feel pretty accomplished and happy. In contrast, if TD does not work well, they start regretting, or morale goes down.

Sometimes feeling after introducing technical debt is subjective.

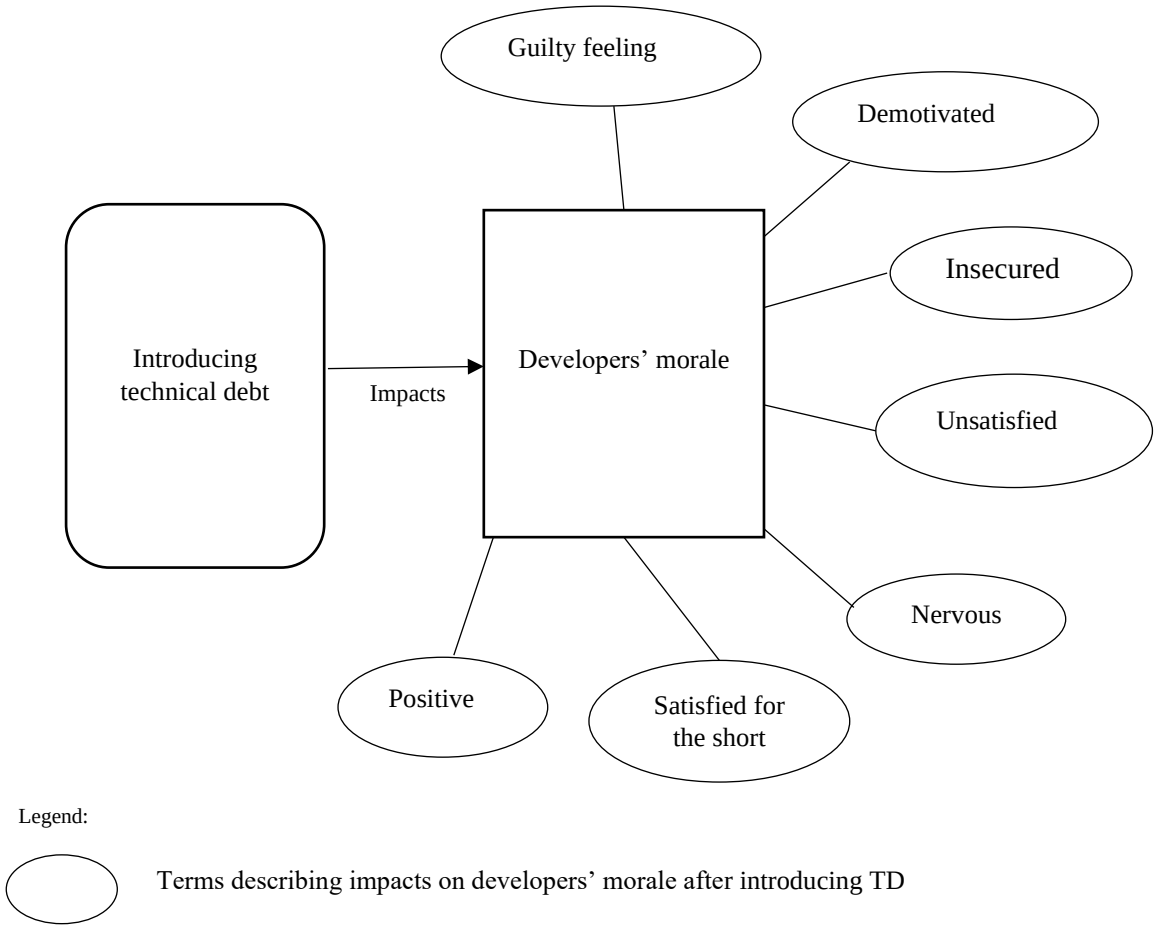


Figure 6: Impacts of introducing technical debt on developers' morale

The above figure 6 summarizes the impact of introducing technical debts on developers' morale. Oval shapes represent impacts on developers' feelings or morale after introducing technical debts.

5.1.2 The impacts of developers' morale after introducing TD on their productivity

Software development is a socio-technical aspect, and software developers are the social aspect. As software developers are also human beings, morale plays a role in their productivity. Apart from a few, majorities of the interviewees thought technical debt lowers their morale down and that that low morale diminishes the performance. If developers understand the nature of debt correctly and are able to maintain peace and focus within themselves, then the adverse effect of technical debt is more diminutive. In contrast, if developers feel frustrated because of TD, their morale drops drastically, ultimately

downgrading their productivity. For example, an interviewee mentioned, *"Since my morale goes down after introducing TD, I think it will reduce my productivity."*

TD lowers developers' morale, and that low morale diminishes their productivity as well.

Thus the productivity of any developers is related to their morale. If introducing TD helps developers to accomplish the task, then their morale will be high and high morale and sound mind increase their productivity. On the contrary, it might be the opposite when technical debt does not work well. For example, one of the interviewees mentioned, "*I think productivity depends on how technical debt is being introduced. If it is introduced in such a way that you will have to work again, it can be a difficult process. You have to learn hard, and then obviously morale and productivity will be down.*"

Developers' morale and productivity are directly proportional: high morale produces high productivity while low morale generates low productivity.

As I discussed in the previous section 5.1.1, most developers do not feel good after introducing technical debt. That bad feeling affects their morale negatively, and low morale reduces their productivity. So, the productivity of a developer might be decreased because of the occurrence of technical debt.

In general, the occurrence of TD decreases developers' productivity.

However, a few interviewees think that their morale goes up after introducing TD because they feel benefited or accomplished in the short term. So, good or high morale motivates them to do more work with more enthusiasm, leading to better performance. For example, an interviewee remarked, *"I think I can perform better after introducing technical debt, and more or less when I am happy, I can write code faster."*

Few developers can perform better after introducing TD in the short run.

So, the productivity of the developer is directly proportional to the developers' morale. If developers' morale increases after introducing TD, then it helps to improve their productivity as well. However, majorities of interviews data reflect that the introduction of technical debts negatively affects the developers' morale. So, from here, what we can draw is TD lowers developers' morale and productivity with some exceptions (might increase the productivity in

the short run. For example, help to complete the project within deadline).

In summary, TD lowers developers' morale and productivity.

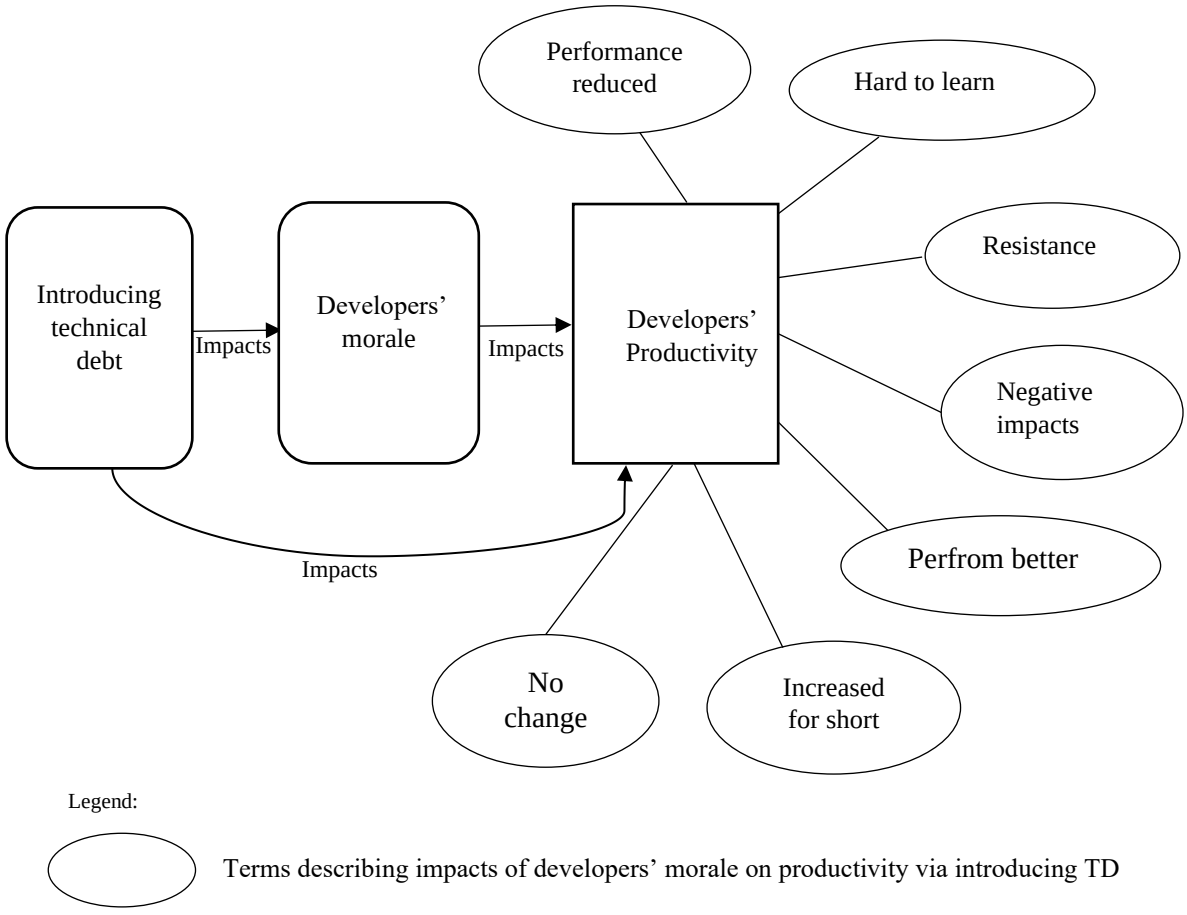


Figure 7: Impacts of developers' morale after introducing TD on developers' productivity

Above figure 7 summarizes the impacts of developers' morale after introducing technical debt on developers' productivity. Oval shapes represent the different impacts on developers' productivity after introducing technical debt.

5.1.3 The impacts on developers' morale when they get criticized for introducing TD

On asking a question, "how do you react when you get criticized for violating quality standards or introducing technical debt?" Most interviewees said they do not have the culture of criticizing each other within a team immediately if they make small mistakes or violate the quality standards. However, they still have experiences of getting criticism or feedback from the upper management level, review team, or maybe from owners or customers sometimes. For example, one of the interviewees mentioned, "*First of all, that situation does not often*

happen to the company I work for now and for the companies I worked for previously. However, sometimes there might be a situation which often happens that product manager and the owner might not be satisfied with the work done, or the design specs do not match the actual implementation of the work."

Usually, within a team, there is no culture of criticizing each other.

The impact of criticism on developers' morale depends upon what kind of criticism it is. Data collected from the interviews suggest that if the criticism is direct, in public, personal, strong, or pointed, it harms developers' morale. An interviewee remarked, *"I think it is quite not good to get criticism in public, but it depends on what kind of criticism it is. If somebody is just pointing out your negative aspects, then you might feel shame, and morale gets down."* So strong and straight criticism demotivates developers; sometimes, they might feel very bad, ashamed, or might not feel good going to work for hours or days.

Personal, pointed, and public criticism negatively impacts the developers' morale.

On the other side, criticism can be a source of motivation to increase developers' morale. If criticism is constructive and given privately, it helps developers to find out their weaknesses, improving their ability to work in a standard way. So, such kind of constructive criticism helps to increase their morale. Likewise, if criticism has been given as feedback with reason and possible solutions, it might not affect their morale. Most of the interviewees said that criticism could be a source of motivation out of which they can improvise themselves and a learning process if it has been given nicely, like feedback in private. For example, an interviewee mentioned, "*If I am getting feedback in the sense that, hey, this piece of code here, what you did here is suboptimal or maybe this could have done better like this or like that. I will gladly take it because I will take it as positive. I have learned something out of it."* Even though developers become a bit more conscious or have the voice of that criticism inside their minds, most of the interviewees will not end up being criticized for that again. It means any sort of criticism that comes along with developers is a new source out of which they can learn.

Criticism could be a new source of motivation and the learning process if it has been given nicely.

When a colleague in a team gets criticized

Software companies usually make a team consisting of different roles from different domains to develop software. So, I was interested to know how a team member reacts if another member within a team gets criticized for violating quality standards. Upon asking a question, "how do you react when your colleague gets criticized or penalized." Most interviewees shared their experience as a negative impact. The whole working atmosphere becomes more concerned and conscious If someone within a team gets criticism. All the members of the team start to think they could be the next person to be criticized. So, other team members might also have a kind of fear or mental pressure all the time. For example, one interviewee remarked, "*I think if a team member gets criticized, then somehow it will be affecting the other team members also. The working speed will be reduced. Other members might also have some kind of fear that I could be the next person.*"

In contrast, few developers do not feel that much pressure when their colleagues get criticized; instead, they try to help them to improve if required. Nevertheless, developers become more conscious about the quality while committing their code next time.

If a colleague in a team gets criticized, it negatively affects others' morale within a team.

Figure 8 below summarizes the impact of criticism for introducing TD on developers' morale in a few relevant terms.

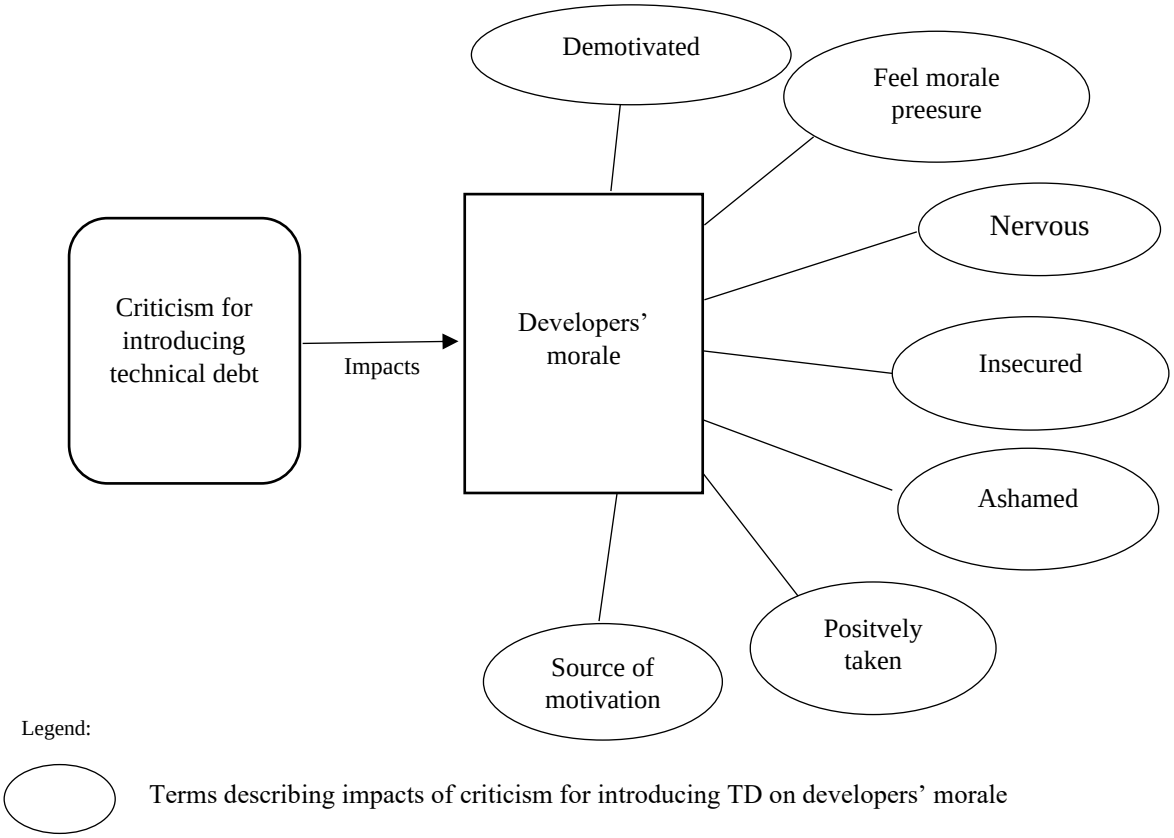


Figure 8: The impacts on developers' morale when they get criticized for introducing TD

5.1.4 The impacts on developers' productivity after being criticized

Direct and pointed criticisms do not help the developers' morale. Developers usually feel demotivated if they get negative feedback or pointed comments on their work. Even though developers mentioned that criticism usually lowers their morale, but when it comes to productivity, it helps. Most interviewees said that criticism does not harm productivity because it is a new source from which they can learn new things, and they are made to be learned. For example, one interviewee said that *"At least as far as I am concerned, criticism helps me to increase my productivity."*

However, few developers think pointed criticism reduces the velocity of the work. Neither the output of work can be good, nor developers can give a quality of work when they get criticized. One interviewee mentioned that it would be the same if a team member within a team gets criticized; the working speed will be reduced somehow.

In conclusion, criticism helps to improve developers' productivity if it is given as feedback. However, some pointed and personal criticism reduces the developers' working speed.

5.2 The impact of management of technical debt on developers' morale and productivity

This chapter explains how different aspects related to the management of TD impact the developers' morale and productivity. Four different aspects: Review, Repaying TD, Criticism, and Team collaboration were studied to see the results.

The impacts of the review process

This section describes how the review process impacts the developers' morale and their productivity. There is not much research in the past to study the impact of the review process (including quality check, code review, and the way developer works) on developers' morale. So, the finding of this section is quite interesting. The impact of code review has been studied in depth.

5.2.1 The impacts of the review process on developers' morale

In order to study the impact of the review process on developers' morale, I asked a few questions to interviewees related to the review process. In responses to those questions, most participants have said that their company has some guidelines and quality standards, except a few. They also have either quality assurance or a review team to inspect; however, sometimes their colleagues or senior reviews the work.

In this regard, most of the interviewees mentioned that they have a positive or a good feeling about the review process. For example, one of the interviewees said, "*It feels good because a human can make many mistakes, and it is good to have another pair of eyes to look in my code then comments on many serious things that I have done so far."* Some of the interviewees take the review process as a way for improvement. They believe that being reviewed by others can be helpful to learn new things or improve themselves.

The review creates a good feeling that helps developers to improve.

Interestingly, some interviewees said that they used to feel morale or psychological pressure at the beginning of their careers. However, over time, they do not feel any kind of such pressure. For example, one of the interviewees said, *"I used to feel nervous or used to have some kind of psychological pressure in the early days of my career, but over time [.....] I do not feel any pressure now."*

Experienced developers feel less pressure on being reviewed and vice versa.

In contrast, few of the interviewees somehow feel nervous or uneasy when they get reviewed. However, they do not take the review process as a psychological or morale pressure, neither they have any bad experience with it. For example, one of the interviewees shared his experience as *"I remember I was kind of nervous, but it was really nice. I had no bad experiences with it."*

Sometimes review makes developers nervous.

Based on this discussion and data collected from the interviews, we can say that the review process positively impacts developers' morale or does not negatively affect their psychology and morale. Even though the review does not have any psychological or morale pressure on the developers, it sometimes makes them a little bit nervous. The study shows that the experience of a developer is an important factor to be considered while measuring the impact of the review process. Developers who have been experienced with software development process for many years usually feel no mental or psychological pressure when they get reviewed. On the other hand, developers with less experience comparatively feel more nervous or pressured to be reviewed.

Most of the developers take the review process positively, which boosts their morale.

Figure 9 below summarizes how developers feel when they get reviewed in a few specific terms.

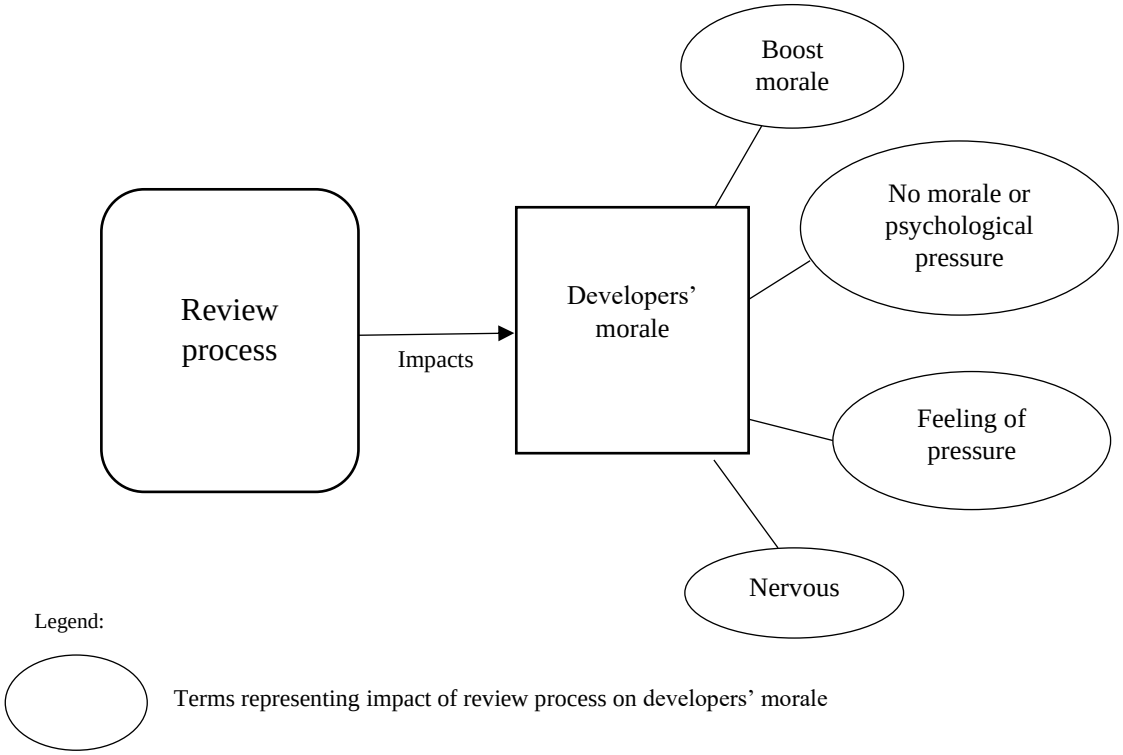


Figure 9: The Impact of the review process on developers' morale

5.2.2 The Impacts of the review process on developers' productivity

Figure 9 below summarizes the impact of the reviews process on developers' productivity with the help of data collected from interviews. Firstly, the impact of the review process on developers' morale was studied (presented in section 5.1.1). Then only the impact on productivity was analyzed based on the morale (which was caused by the review process).

As mentioned earlier review process positively impact the developer morale even though sometimes developers feel a bit nervous or psychological pressure. Most interviewees mentioned that although they feel nervous, psychological, or morale pressure on being reviewed, reviews help to improve themselves. It helps to improve their coding level, maintain the overall software quality, or learn new things. For example, one of the interviewees said that *"Obviously, if someone is reviewing, it will feel a little bit bad or nervous. However, the review process is a good thing to improve yourself."*
So from the interviews data, it is pretty clear that the review process somehow might affect developers' state of mind for a short instant. However, they think the review is a good thing to improve quality, their coding level, or their working capabilities in the long run which

ultimately can help the developers' productivity. So, the review helps developers' productivity.

In general, the review helps to improve software quality and developers' productivity.

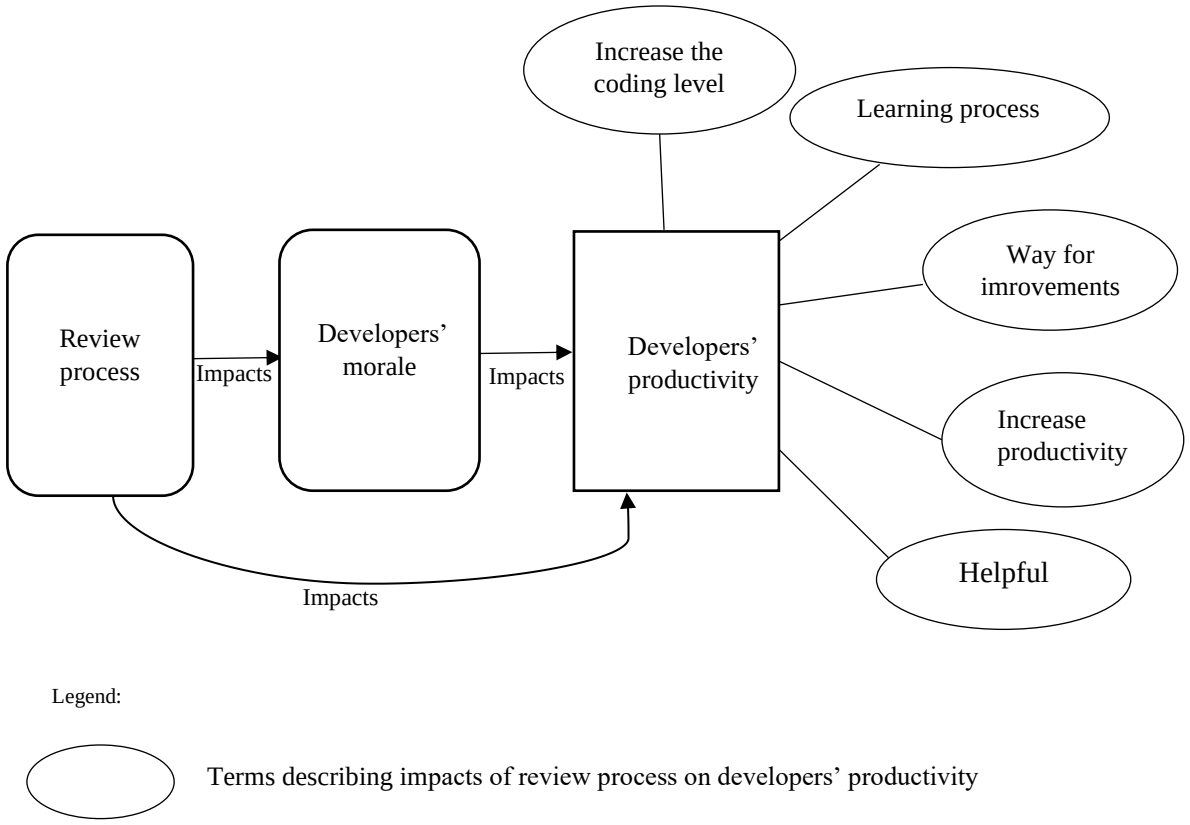


Figure 10: The Impacts of the review process on developers' productivity

The above figure 10 summarizes the impact of the review process on developers' productivity in a few terms. Oval shapes represent the impacts of the review process on the developers' productivity.

5.2.3 The impacts of repaying the technical debt on developers' morale.

Except few, most of the technical debt should be paid at some stage of the software development process. So, this section will discuss how developers feel what developers feel while and after repaying technical debt. Firstly, we discuss the impact of technical debt on developers' morale where the debt was introduced by the respective developer who is repaying now. Later, we will explain developers' feelings on repaying TD, where TD was created by someone else.

Developers' feelings while repaying TD

On asking a question, "how do you feel when you need to spend your time to repay the TD that you have accumulated in the past?" most interviewees reflected it as negative feelings. They mentioned that repaying TD is frustrating since it takes longer to understand what was done before and what needs to be fixed now. So, repaying technical debt could be a mental pressure to developers because on one side, there will be many things to rectify, and on another side, it keeps piling up the task you need to finish because of the debts. Developers believe that repaying TD increases time and resources, which obviously makes negative impacts on their morale. For example, one interview mentioned, *"Spending more time and resources obviously makes a negative impact on your mind."*

Repaying TD is a frustrating task for some developers, which lowers their morale.

Spending time on repaying TD creates a sort of fear of being in a hurry to do regular stuff like implementing new features. So, such kinds of fear of being lagged in the software development process negatively affect the developers' psychological thinking, or they might not be able to think in the big picture at that moment. Nobody wants to be unproductive or less productive by doing the same work repeatedly. So, the level of motivation of any developer will be reduced when they have to spend time repaying the technical debt. For example, one interviewee mentioned, *"Nobody wants to do the same work again and again. The level of motivation you had at the beginning of work will obviously be reduced when you have to spend time on repaying debts. As a result, it will be difficult to complete the task properly on time.*

Developers' level of motivation will be reduced.

Even though some developers accept that repaying TD is a problematic situation to be in, they would not necessarily feel that much pressure in that regard. Thus, it is a kind of okay feeling. An interviewee mentioned that *"I would just understand that as this is the current problem that we have, then I will just try to fix it. So, I will by no means stop crying [….] all these old techs and all that."* More or less, developers think that they were responsible for introducing TD, so they assume the process of repaying TD as their responsibility without any mental or morale pressure.

Some developers expressed their different perspectives, where repaying TD does not negatively impact their morale. Developers do not feel any psychological or mental pressure because they were already aware of such situations while introducing TD. So, developers easily accept and embrace whatever comes because they already knew that this would happen in the future. For example, an interviewee said that *"developers need to pay TD later. Normally, we are ready to pay those debts in such situations because somehow, we were aware of this. At that time, we do not feel that much guilt and feel pressure."*

Some developers think differently as repaying TD is also a part of work that they easily accept and embrace without any psychological or mental pressure..

Repaying technical debt created by others

The software development process is teamwork. There might be a group of people with different roles. Sometimes, some situations may arise where one needs to pay the technical debt accumulated by the other. After analyzing the interviews data, developers psychologically do not like to be involved in fixing the debt created by others.

Developers usually do not like to be involved in solving the debt created by others.

One interviewee mentioned, "*I do not feel good if someone creates debts and I am supposed to pay. I want to avoid it as much as possible if permissible by my position*." However, it depends on the situation. If someone created the debt because the situation while creating debt was business-critical and that TD was the only way to solve those issues, then everyone in the team would happily love to help. Otherwise, if someone created debt just because they like the specific tech or way to solve the task, then other colleagues in the team probably do not like to be part of this.

If the debt was created because of some business-related critical situations (like delivery before the deadline) then developers want to be a part of it.

The developers' feeling on repaying TD created by others is a bit subjective. If the TD affects a lot or needs lots of extra work and time to fix, developers feel bad or pressured. It all depends on how much time and resources are allocated for it. A short period of time and limited resources to fix debts often creates more pressure because it might take longer to understand. For example, an interviewee said, *"paying TD is a mental pressure either way. If debts were created by somebody else, then it might take much time to understand what was done and gets frustrating."* If developers spend much time fixing technical debts, they usually panic and become frustrated and want to come out from such situations and perform their regular tasks.

There are other different perspectives where some of the developers happily want to help their colleagues as they are working in a team. For example, one interviewee mentioned, *"For me, it does not matter who created the debts since you are working in a group for a company, I would not mind helping on solving the debts."*

Some developers feel happy to solve those debts as they are working in a team.

The developers' feeling after repaying technical debt

Paying technical debts in itself is a sense of achievement and relief to the developers, and it is common to be praised or appreciated for fixing debts. If developers fix the debts, they feel more satisfied because they have learned how to do better and feel like they have improved themselves. Such kinds of self-satisfaction even might not need praises from others to uplift their morale. Repaying technical debt successfully helps developers to come out from the early stage pressure of technical debt that makes them feel satisfied and motivated. Along with this, colleagues from the team also appreciate it. In this regard, one interviewee responded, *"I mostly get compliments like good work since I have finished something but not like in details on what they liked about it. They always appreciate and praise. it improves my overall mental health and as long as it plays good roles in my work."*

Develoeprs feel satisfied and motivated after repaying TD.

So, almost all interviewees agreed that they feel satisfied, motivated, appreciated, or praised after repaying technical debts. They feel like that they have achieved something they have been looking for for a long time. In contrast, one interviewee mentioned that paying TD makes developers feel satisfied but does not create chaos in a broader sense because they already knew about the situation where they need to acknowledge debts that they have introduced in the early stages.

Develoeprs feel satisfied after repaying TD but not very happy.

Below, figure 11 illustrates the impacts of repaying TD on developers' morale by a few specific terms.

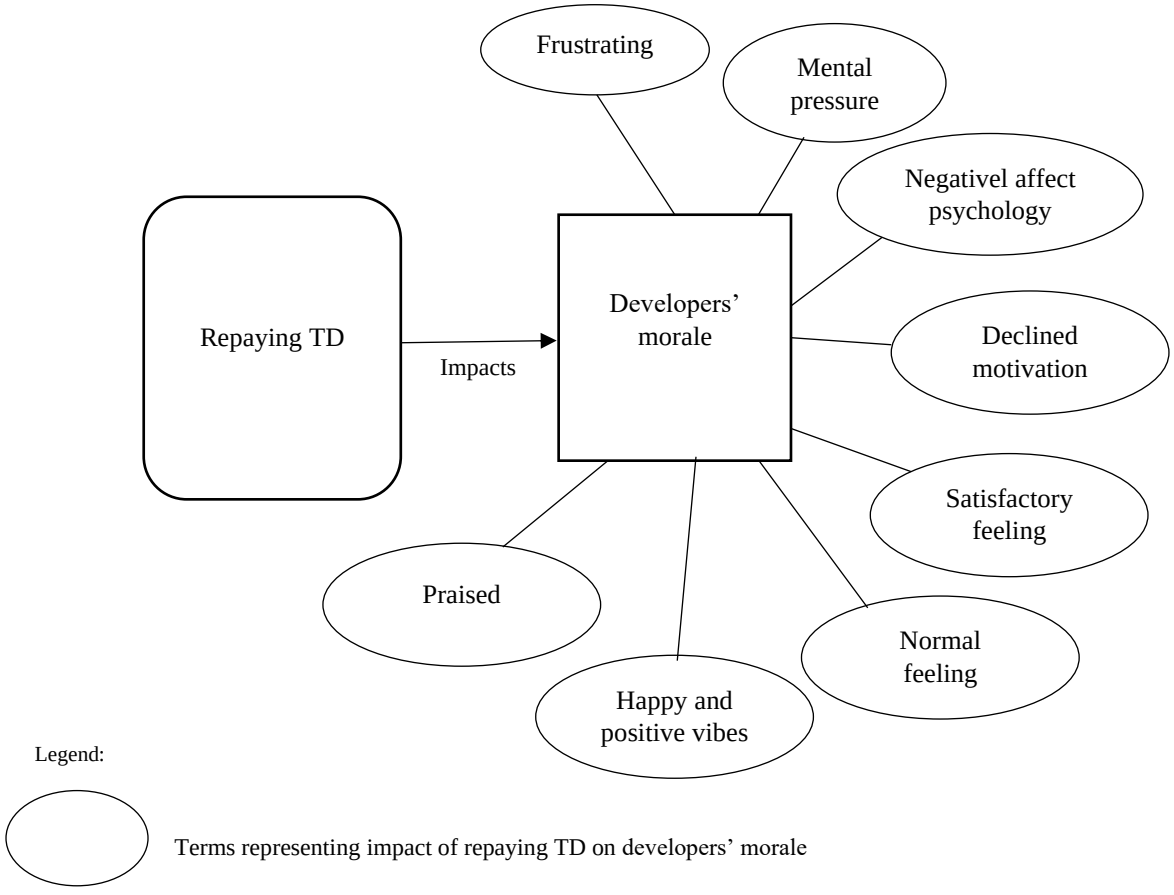


Figure 11: The impacts of repaying TD on developers' morale

5.2.4 The impacts of repaying technical debts on developers' productivity.

In general, developers do not feel good if they need to refactor technical debt. Repaying technical debt usually lowers down the developers' morale which ultimately reduces their productivity. In order to pay technical debts, first, developers have to learn a lot more about what has been done, what is next to be done, and how to fix those things. It might take much time to understand the process, which makes developers panic or frustrated. So, the process of refactoring technical debt initially decreases developers' productivity. If there was no debt, developers could have delivered what was required within the deadline, but repaying technical debt slows them down and can resist making progress. For example, one interviewee mentioned that *"It slows you down or like you feel lagged in time, because if there were no debts then you could have exceeded that point in time and delivered what was required. It feels like time has slowed down, and you are not making any progress. So yeah, it does kind of affect the productivity."*

Repaying TD with low morale slows developers down and resistance to making progress.

Developers may have to work on something else instead of focusing on what they need to develop, such as building new features, feature on top of it. So, paying TD often affects productivity negatively, and a new feature will not be delivered on time. In other words, technical debt might bounce you back in the process of developing software.

Most interviewees mentioned that paying technical debts reduces productivity in as sense that it increases the required resources (such as time, extra developers, cost). For example, one interviewee said, *"It depends on the debt how high the debt is. If the technical debt brings you to rework everything, it will take obviously increase the time and resources which ultimately reduces the productivity."*The assigned tasks within the sprint will not be completed if someone is fixing TD, whose morale has been down. So, this overall process of paying debt with low morale may increase the cost, and the allocation of resources might be imbalanced. For example, one said, *"I think it affects the time, resource, and cost because you will take much time to understand and fix the debts. You are using lots of resources on that which usually leads to higher cost of the product, and since you are fixing the debts, you will need more time and resources than previously given."*

Repaying TD with low morale increases the required resources and reduces productivity.

On the other side repaying technical debt can boost up some developers' productivity. If those things (debts) work properly, it increases their morale which helps developers to complete the work faster. Developers sometimes feel more satisfied because repaying technical debts is a sense of achievement as those tasks are long overdue which need to be fixed. So, repaying TD might help developers to deliver the product. They might feel like they had made some contribution to the project, which helps to boost their performance. Besides, repaying technical debt provides room for improvement since it gives an opportunity to learn and handle such pressurized moments.

Sometimes repaying TD boost up developers' productivity and provides room for improvement.

Surprisingly, one interview said that repaying technical debt does not affect productivity who mentioned, *"Solving technical debts is also a kind of gaining a new learning experience. So it might not affect the productivity for me at least but, it depends on how big the technical debt is."* While next mentioned, *"I will have same involvement. I think repaying TD does not change anyway."*

For few developers, repaying TD does not affect their productivity.

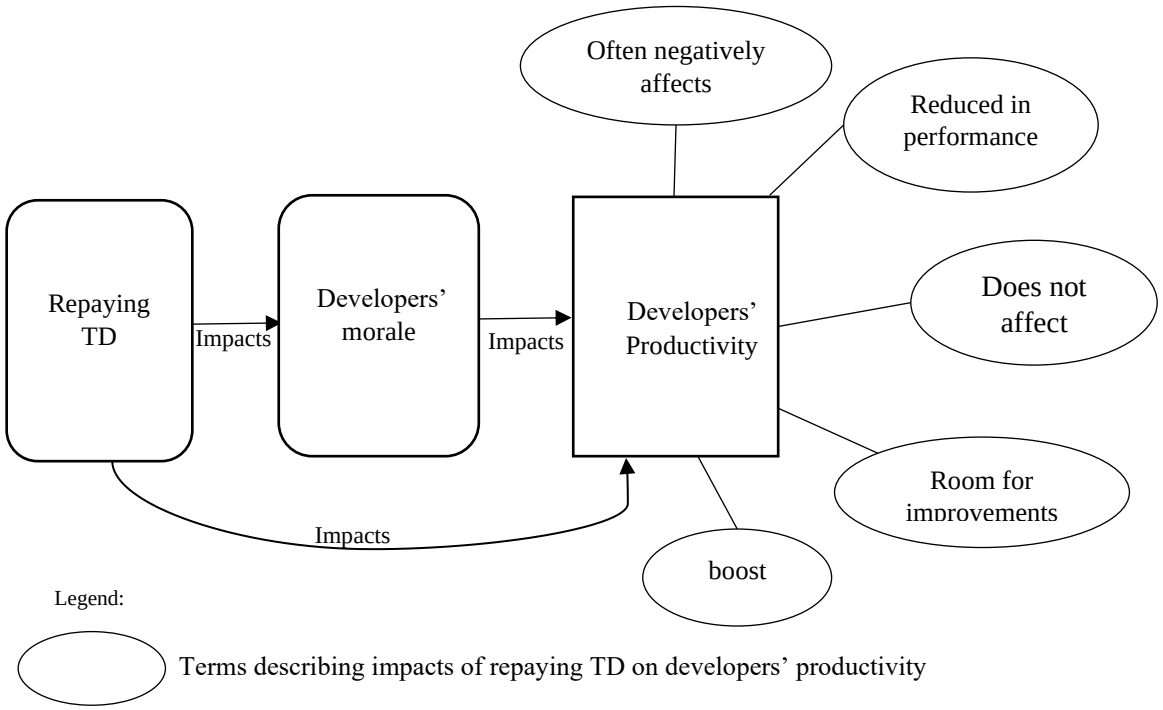


Figure 12: The impacts of repaying TD on developers' productivity

5.2.5 The impacts of praise for repaying TD on developers' morale

Praise always has a good impact on developers' morale. One developer said, *"if I am performing almost 70/80% and get some praise, then it boost up my morale, will try to perform better than before, and I will pay more attention."* When a developer has fixed the debt, there will be a moment of the entire team being in a happy mood. They praise the developer who has paid the debt, and that praise makes the developer feel good. For example, one interviewee said, *"Being praised will make you feel good and also makes your mood happy. What I believe is if you fixed anything like bugs, debts, or features, then you will feel very good and relief."*

Most interviewees mentioned that they have a culture of praising each other if someone has paid technical debts or any other issues in the development process. Developers believe that if they are able to pay the technical debt, they get appreciated or praised most of the time, which makes them feel happy, sound, or more satisfied. For example, an interviewee remarked as *"I mostly get compliments like good words since I have finished something but not like in details on what they liked about it. They always appreciate and praise. it improves my overall mental health and as long as it plays good roles in my work."*

Praise increases the developers' morale and improves mental health.

Software companies have a culture of cheering up teammates once they fix or pay debts. They often applaud and cheer to make developers happy and proud of solving those overdue debts. The culture of appreciating each other makes the whole working environment happy, motivated and provides good vibes. So, it is always good to appreciate, acknowledge and celebrate each others' contributions that increase the team morale.

Praise makes the working environment happy and increases the overall team morale.

5.2.6 The impacts of praise for repaying TD on developers' productivity

Praise is an essential source of developers' motivation and high morale. Almost all the time, praise increases the developers' morale which definitely helps the performance. When developers get praised, enthusiasm will be raised that motivates them to work more sincerely and actively. Praise boost morale which helps developers to work better than before.
For example, one interviewee said, *"praise or appreciation increases the working speed of*

developers." Praise spreads positivity in the team, which often has a good impact on developers' performance.

If any developer gets praise for fixing technical debts, it increases their responsibilities as well. They pay more attention to their work, which ultimately helps to achieve qualitative work. So, praise always makes the developers feel special and more responsible because everyone will expect the same workability next time. Thus, all these things help developers to increase their productivity.

Praise helps developers to increase their productivity.

5.2.7 The impacts of team collaboration for fixing technical debt on developers' morale

Impact on developers' morale if colleagues help to fix TD

The most important thing while working in a team is team communication and connection. One interviewee believes that if a colleague helps, developers feel more connected with the team, which allows them to do more. Most team members in a team are fond of team collaboration. For example, an interview said, *"I like being helped or have a friendly tone when you can, like, see what's and no negativity."* The developers feel pretty good about having a good team collaboration or teammates who can pinpoint what needs to be done to address the debts and fix them. So, most developers feel encouraged and happy, and they feel like they are also a part of that team.

Team collaboration encourages and makes developers happy.

On asking a question, "how do you feel when your colleague helps to fix the TD?" to 11 interviewees, all of them expressed their positive feelings about team collaboration. They said team collaboration brings them more close and binds them as a concrete team. Most of them often ask for help and usually receive it from co-workers, making them thankful and happy. Having another pair of eyes, extra skills, and knowledge to fix the debts would be helpful, and most developers appreciate such kinds of help from the team. Team collaboration will reduce the cost of all the mental pressure. For example, one interviewee mentioned, *"if colleagues help fix the debt, then the cost of all the pressure will be reduced for me. So, I feel happy because the cost of the product will significantly decrease in that case."*

Team collaboration spreads positivity among the team and reduces developers' psychological pressure.

Impact on developers' morale if colleagues do not help to fix TD

If team members do not help each other to fix technical debts, the teamwork will decline and might go even worse. Some interviewees said that not getting help from the college is frustrating. Developers prefer to have at least someone who is helpful and more knowledgeable in their ream. So, no one will feel good if their colleagues do not collaborate with them. Since most developers love to consult with their team members, it might demotivate them if they do not receive help.

No developers feel good or get motivated if there is no collaboration in the team.

Sometimes, colleagues might not have any idea how to help or be busy with their own priorities. In such situations, it does not negatively impact developers' morale even though they did not receive help. However, if colleagues did not help despite knowing the solutions, it harms the developer morale and makes coordination lose. For example, one interviewee said, *"if I know that they have the solutions and do not help me, it can be quite frustrating and bitter experience in the team."*

There is no morale impact if the colleagues do not know how to solve the TD or do not have enough time to help even though they want to help.

So, not having proper team coordination to fix the debt creates extra mental or psychological pressure on developers. Developers sometimes feel like they have made mistakes by accumulating technical debt. They feel like a burden to themselves if somebody in the team is not ready to help them. So, if there is no teamwork, coordination, or communication, it drastically diminishes developers' morale.

Lack of proper team coordination creates more psychological pressure and decreases developers' morale.

5.2.8 The impact of team collaboration for repaying TD on developers' productivity.

Since everyone is working on a team, collaboration is essential from the perspective of developers' morale and productivity. Team collaboration will help increase the overall teams' productivity as it boosts developers' morale most of the time. If anyone in a team is slower or less productive while refactoring technical debt, they should be helped by another team member to increase productivity.

Most interviewees mentioned that team collaboration creates a good and positive vibe in the team, and it increases the team connection while ultimately increasing productivity. On the other hand, team collaboration makes it easier to fix the technical debt and helps finish the software project within the deadline.

So, with all these positivities, good environment, proper team connections, extra hands to help good feeling, and high morales, developers get motivated to do the work correctly with enthusiasm. Developers pay extra attention and concentration that allows them to find out the idea to fix the overdue technical debts and finish the software within deadline and predetermined resources. Thus, team collaboration increases the developers' productivity.

Team collaboration spreads positive vibes, which increase developers' productivity.

5.3 The summary of findings from this study

This section explains the summary of findings from sections 5.1 and 5.2. These findings are based on the interviews data. Different subsections described above represent how different aspects related to the occurrence and management of TD affect the developers' morale and productivity. The impacts of aspects related to the occurrence of TD are presented in the first two columns of table 5 (i.e., S.N1 and S.N.2). In contrast, the other four columns from S.N.3 to S.N. 6 summarize the impacts of management of TD on developers' morale and productivity.

Below, Table 5 summarizes the finding of this thesis concerning two research questions.

Table 5: Summary of findings of this study

S.N.	Aspects related to the occurrence of TD	Impacts on developers' morale (RQ1)	Impacts of morale on developers' productivity via the occurrence of TD (RQ2)
1	Introducing technical debt	- Usually, introducing TD negatively impacts developers' morale and demotivates them. - Technical debts can provide short-term happiness or satisfaction for a few developers. - Sometimes feeling after introducing technical debt is subjective. If TD works well, developers feel good; otherwise, their morale goes down.	- TD lowers developers' morale, and that low morale diminishes their productivity as well. - Developers' morale and productivity are directly proportional: high morale produces high productivity while low morale generates low productivity. - In general, the introduction of TD decreases developers' productivity. - Few developers can perform better after introducing TD in the short run.
2	Criticism for introducing technical debt	- Usually, within a team, there is no culture of criticizing each other. - Personal, pointed, and public criticism negatively impacts the developers' morale. - Criticism could be a new source of motivation and the learning process if it has been given nicely. - If a colleague in a team gets criticized, it negatively affects others' morale within the team.	- In conclusion, criticism helps to improve developers' productivity if it is given as feedback. However, some pointed and personal criticism reduces the developers' working speed.

S.N	Aspects related to the management of TD	Impacts on developers' morale (RQ1)	Impacts of morale on developers' productivity via management of TD (RQ2)
3	Review process	- The review process creates good feelings that help developers to improve. - Experienced developers feel less pressure on being reviewed and vice versa. - Sometimes review makes developers nervous. - Most of the developers take the review process positively, which boosts their morale.	- Even though developers feel nervous or pressured, the review helps to improve the developers' productivity. - In general, the review helps to improve software quality and developers' productivity.

4	Repaying technical debt	- Repaying TD is a frustrating task for some developers, which lowers their morale. - Developers' level of motivation will be reduced. - Some developers think differently as repaying TD is also a part of work that they readily accept and embrace without any psychological or mental pressure. - Developers usually do not like to be involved in solving the debt created by others. - If the debt was created because of some business-related critical situations (like delivery before the deadline), then developers want to be a part of it. - Some developers feel happy to solve those debts as they are working in a team. - However, developers feel satisfied and motivated after repaying TD. -Few developers feel satisfied after repaying TD but not very happy.	-Repaying TD with low morale slows developers down and resistance to making progress. - Repaying TD with low morale increases the required resources and reduces productivity. - Sometimes paying TD boost up developers' productivity and provides room for improvement. - For a few developers, paying TD does not affect their productivity.s
5	Praises for fixing technical debt	- Praise increases the developers' morale and improves mental health. - Praise makes the working environment happy and increases the overall team morale.	- Praise helps developers to increase their productivity.
6	Team collaboration for fixing technical debt	- Team collaboration encourages and makes developers happy. - Team collaboration spreads positivity among the team and reduces developers' psychological pressure. - No developers feel good or get motivated if there is no collaboration in the team. - There is no morale impact if the colleagues do not know how to solve the TD or do not have enough time to help even though they want to help. - Lack of proper team coordination creates more psychological pressure and decreases developers' morale.	- Team collaboration spreads positive vibes, which increase developers' productivity.

6. Discussions

This section will discuss the results presented in the previous chapter 5. Along with this, my interpretations and opinions will also be presented. Throughout the discussions, research questions will be answered at first with the help of results and collected data, followed by a discussion on the contribution of this study towards the human aspects (specifically developers' morale and productivity) of technical debt. This section will also compare and contrast the results with previous studies and try to find out what can be new outcomes in this research field. Threats, validity, and limitations of this study will also be presented, followed by the mitigation practices.

This study mainly has two research questions: the impact of occurrence and management of technical debt on developers' morale and the impact of developers' morale on developers' productivity via occurrence and management of TD. The impact of the occurrence of TD on developers' morale and productivity will be discussed in section 6.1, followed by the discussion about the impacts of different aspects of TD related to management on developers' morale and productivity in section 6.2. Additionally, with the results, these sections will compare and contrast the results with relevant literature.

6.1 The impact of the occurrence of TD on developers' morale and productivity

From the collected data and the previous studies (Tom et al., 2013), (Besker et al., 2020), (Ghanbari et al., 2017), (Besker et al., 2019), (Yli-Huumo et al., 2014), it has been cleared that different aspects related to the occurrence of TD have impacts on developers' morale and productivity directly or indirectly. So, I will discuss how introducing TD and criticism for introducing TD affect developers' morale and productivity individually. Along with the results, this section will compare and contrast the results with relevant literature.

6.1.1 Introducing technical debt

Impact on developers' morale after introducing technical debt (RQ1)

Few studies look into the impact of technical debt on developers' morale, but they mostly explain it in general. It means those studies did not explain the impact of different aspects of technical debt independently in depth. So, discussing the impact of introducing technical debt on developers' morale alone is itself a more specific finding than related studies.

While there has been considerable interest in studying TD in recent years, very few studies examine the phenomenon from the perspective of individuals, particularly those that focus on the influence on developer morale. A minimal number of studies (Ghanbari et al., 2017), (Besker et al., 2020), (Tom et al., 2013), (Yli-Huumo et al., 2014), (Fernández-Sánchez et al., 2015) suggest that technical debt may affect individuals' emotions (significantly developers' morale). Likewise, findings of this study indicate that developers usually do not feel good after introducing technical debts, which leads to dissatisfaction with their own work, negatively impacting their morale. This statement is supported by a case study (Yli-Huumo et al., 2014) which referenced an interviewee answer that taking technical debt does not feel good to take from developer's point of view because they know that the solution is not the best one and they might have to fix it later. Developers generally prefer to have no technical debt and just want to build "perfect software" because they must interact with the code base on a day-to-day basis. As a result, developers are more aware and concerned about potential difficulties that may occur after introducing technical debt (Lim, Taksande, & Seaman, 2012). Cutting corners or introducing TD by disregarding the standard development process also lowers the developers' morale and motivation and reduces one's self-esteem (Peters, 2014).

It is worth noting that morale after introducing technical debt is subjective. If the technical debt performs as intended by the developer, they might feel good and satisfied since they complete the project within the deadline or predetermined resources. So, contrary to some studies (Yli-Huumo et al., 2014), (Besker et al., 2020), (Tom et al., 2013), technical debt can occasionally provide short-term delight or satisfaction, which enhances developers' morale and self-esteem despite their awareness of potential negative consequences. However, if the concept of introducing technical debt fails to achieve the expected outcome, the situation may deteriorate, leaving developers disgruntled or dissatisfied.

Impact of developers' morale after introducing TD on productivity (RQ2)

The relationship between software developer happiness and work-related characteristics such as performance and productivity has been highlighted in recent research in the field of behavioral software engineering (Graziotin et al., 2018). Developers' creativity, self-confidence, and productivity increase when they are happy. On the other side, unhappiness creates negative consequences on developers such as low productivity, broken working flow and also compels the developer to take short corners (compromising qualities, creating

technical debt) (Graziotin et al., 2018). So, happiness can be taken as a productivity and performance booster for developers, while unhappiness has adverse effects. Respondents in a study (Besker et al., 2020) mentioned that the developers' progress and productivity might be hindered by technical debt.

A study by Ghanbari et al. (Ghanbari et al., 2017) indicates that there is a stronger correlation between employees' morale and productivity at work. In line with this statement, the findings of this thesis suggest that technical debt usually lowers the developers' morale and the developer with low morale mindset diminishes their working speed, flow, and their productivity. However, introducing TD sometimes brings joy, enthusiasm, and motivation towards the work in the short run when developers feel accomplished and benefit from TD. In such situations, the developers' productivity might go up.

The results of this thesis show that most of the time, developers' morale, level of happiness, and satisfaction drop after introducing technical debt because they are all aware of the possible consequences of those debts that unknowingly decrease their potential productivity. Thus, introducing technical debt is also a psychological cost that reduces developers' productivity (Laribee, 2009).

6.1.2 Getting criticism

The Cambridge Dictionary (Dictionary, 2021) defines criticism as "*the act of giving your opinion or judgment about the good or bad qualities of something or someone*." If the introduced technical debt creates problems in the software development process, developers are likely to get criticism. However, the results of this study show that there is no culture of criticizing team members in most software companies even if they introduce technical debt. This finding matches the results of the study (Besker et al., 2020) that argues that neither developers have been subject to criticism nor blame co-workers for introducing technical debt. Since (Hardy, 2010) said that criticism is a negative affective antecedent of morale, studying its impacts on developers' morale makes much sense.

Impact on developers' morale (RQ1)

Since there are not many studies that relate the criticism for taking technical debt with developers' morale and productivity, this discussion is mainly based on the results of this study. The results show that the impact of criticism depends on how it has been given to the developer. Mostly personal, pointed, strong, and public criticism negatively influence

developer morale. Developers usually feel unsatisfied with getting criticism for creating technical debt. However, most developers personally do not want to work in such a way that leads to the occurrence of technical debt (Ghanbari et al., 2017). In contrast, constructive criticism or positive feedback could be a new source of motivation, satisfaction, and learning opportunities that could boost developers' morale.

It is also important to understand the impact of criticism on one developer's morale if another team member gets criticized. Thus, the results clearly demonstrate that every developers' team morale suffers when their colleagues get criticized. This occurs because each team member starts to overthink and cause psychological burden in anticipation of being the next to be criticized.

Impact on developers' productivity (RQ2)

When it comes to productivity, the results imply that the working speed of a software development team reduces because of the low morale and psychological pressure after getting criticism. Criticism for taking technical debt could also affect the software's quality. However, there are few mixed perceptions that criticism could be a source of the learning process if given nicely. It might help to work in a better way, ultimately increasing productivity.

6.2 Impact of management of TD on developers' morale and productivity

The impact of the management of TD on developers' morale and productivity will be discussed based on the following four aspects of TD.

6.2.1 Review Process

Impact of the review process on developers' morale (RQ1)

"Quality is never an accident; it is always the result of high intention, sincere effort, intelligent direction, and skillful execution; it represents the wise choice of many alternatives" (William A. Foster). Errors, bugs, or issues may inevitably occur during the software development process. It is critically essential to acknowledge and solve those issues at the early stage of software development; otherwise, that might compromise the quality and introduce technical debt in the future. Software reviews are usually conducted during and after the software development process to ensure that the product is bug-free and not compromise quality and guidelines (Alsayed, Bilgrami, & Foster, 2017).

Cunningham (Cunningham, 1992) introduced the term technical debt to describe a situation where long-term code quality is traded for a short-term gain. As the review process checks all the quality standards and guidelines, it might be helpful to developers to prevent themselves from introducing TD in the short or long run. So, the review process is somehow related to the prevention and management of technical debt. Sometimes, it helps avoid introducing technical debt, and sometimes it helps to manage technical debt. So, the following section will discuss how the review process impacts developers' morale and productivity.

To the best of my knowledge, very few studies explain the relationship between the review process, technical debt, and its impact on developers' morale. As the review is one of the interpersonal antecedents of morale (Hardy, 2010) and related to technical debt, it obviously could impact developers' morale as well. The findings of this study show that software developers take the review process as an essential aspect of the software development process. The review process usually boosts their morale since it helps them improve on their work to achieve quality. A research study (Ghanbari et al., 2017) also supports this statement, stating that the review is a positive contribution that helps developers identify technical debts and help each other improve software qualities. If we go even more specific, code review is constructive, and it boosts developers' morale. Code review might help find out minor issues or bugs in the code, strengthening the code quality. The excellent quality code protects developers from introducing technical debt, and hence it increases morale. This statement is also supported by a study (Yli-Huumo et al., 2014), which argues that code review helps to reduce technical debt.

Even though most of the time, code reviews helps developers to strengthen their morale, sometimes it makes developers nervous. The level of nervousness depends on the developers' experience. Developers with high experience usually take the review as a routine procedure, but developers with low experience may consider it an additional responsibility, making them apprehensive and stressed.

Impact of the review process on developers' productivity (RQ2)

The review has positive impacts on developers' morale which motivates them to work more, helps to improve the quality of the software, developers' working ability, and productivity. Interestingly, even though the review process sometimes makes developers feel a little nervous or stressed, it helps them increase their productivity because it helps to improve the quality of their work. So, from this finding, we can say that developers' productivity solely

does not entirely depend upon the developers' morale. Sometimes, it might depend on other factors such quality of the code.

6.2.2 Repaying technical debt

Impact on developers' morale (RQ1)

Repaying technical debt is a frustrating task for some developers, which lowers developers' level of motivation and hence morale. Spending time on managing or paying technical debt piles up the developers' other essential tasks, such as feature development and implementations. Not being able to rectify the task on time affects the developers' psychological thinking or creates fear in developers' minds. So, paying technical debt hurts the developers' morale. This is in line with Laaribee (Laribee, 2009) that says that the pain of not being constructive affects developers' state of mind. Interview participants in a study (Tom et al., 2013) mentioned that many tasks to prevent or repay the accrual of technical debt are not enjoyable.

However, in contrast with studies (Tom et al., 2013), (Hardy, 2010), few developers think that paying TD as a part of their regular work who readily accept and embrace without any psychological or mental pressure. But, most developers usually hesitate to solve the technical debts created by others. However, it still depends on whether debts were made because there was no other way to come out and in such situations, developers easily accept paying the debts without hurting their mental health. However, if debts were created without proper reasons just to experiment with new things, developers do not prefer to be part of those TD. At the same time, few developers feel happy to pay the debts as they work in a team even though debts were made by someone else on their own.

Based on the results of this study, even though developers accept that paying TD could be a problematic situation to be in, it does not necessarily create morale or psychological strain. However, this contradicts the finding of (Ghanbari et al., 2017), arguing that developers feel satisfied and happy with repaying technical debt. Additionally, the results indicate that developers feel relieved, satisfied, and motivated after paying technical debts, and morale might increase. This statement is also supported by a study (Besker et al., 2020) that argues that repaying TD improves software quality, and such improvements in quality enhance developers' morale and level of motivation.

However, developers might not be too happy and excited after paying TD because they were well known about the possible consequences at the time of introducing TD. Another reason is that they have already invested extra effort to pay those debts.

Impact on developers productivity (RQ2)

The results show that there is an intertwined relationship between developer morale and productivity. The process of refactoring or repaying technical debt lowers the developers' morale, and hence low morale negatively influences productivity. The reason behind declining morale and productivity is an extra effort required to identify, document, and refactor the accrued technical debt. Developers believe that if there are no debts, they could work freely without any objection and extra work that enables them to finish their task within the deadline. This finding is supported by a study (Spínola et al., 2013) that argues that working off debt is beneficial and motivating that improves morale which has significant positive influences on productivity.

Interviewees in this study reflect that paying technical debt usually increases required resources such as cost, time, number of developers. Due to this reason, the overall productivity decreases where the productivity is defined in terms of work completed within time, cost, and resources. Thus, the results recommend that paying technical debt with low morale slows development and increases resistance to making progress. This result contradicts other research such as (Besker et al., 2020), (Ghanbari et al., 2017), which argues that paying TD encourages developers and enables them to improve quality and their task. So, it is essential to understand if the impact we are discussing is during the refactoring process or after the successful refactor of debt.

On the flip side, the results also indicate that paying technical debt could be a source of developers' motivation if the incurred TD works well. In such situations, developers feel a sense of achievement that leads to higher productivity. This statement is in line with the finding of a study (Damian & Chisan, 2006) that argues that developers' pride as a result of ongoing software enhancement has a beneficial effect on developers' morale and ultimately results in increased productivity.

However, if we discuss about the impact on productivity after the successful refactoring or repaying of technical debt, it could be different. The results show that developers feel satisfied and motivated after paying or refactoring technical debt. However, few developers

could not be pleased even after paying the technical debt because they have already wasted their time and extra resources to repay those debts. Successfully paying technical debt without making any compromises is itself a sense of achievement. Developers usually get an opportunity to learn new technical and non-technical things while refactoring the technical debt that makes them more proud. So, coming out from the pressure of technical debt motivates them to work more qualitatively, which ultimately helps to increase their productivity. This finding matches the findings of other researchers (Besker et al., 2020), (Ghanbari et al., 2017).

As an exception, the results of this study show that paying technical debts barely impacts the productivity of some developers. So, it should be adequately researched to identify what makes the differences not making any impact on productivity.

6.2.3 The impacts of praise for repaying technical debt

Unlike criticism, praise is a positive affective antecedent of morale that increases developer morale (Hardy, 2010). Nonetheless, the role of praise or appreciation for repaying technical debt has also received minimal attention as criticism got. Only a few research have been studied this impact from the surface. So, the role of praise after repaying TD on developers' morale and productivity is mainly discussed based on the results of this study, along with a few previous studies (Besker et al., 2020), (Ghanbari et al., 2017).

The impact of praise after repaying technical debt on developers' morale

As long praise is regarded as sincere, it is especially advantageous to motivation when it promotes performance attributions (Henderson & Lepper, 2002). A software team or a particular developer loves to be praised for managing or fixing the technical debt. The results show that praise plays a vital role in improving developers' mental health and morale. Unlike an interviewees' perspective in a previous study (Ghanbari et al., 2017), which stated that their company lacks the culture of praise and appreciation, the results of this study depict that all of the software companies studied have a culture of appreciating each other for repaying the technical debt. The praise or appreciation instills joy in the workplace, allowing developers to feel sound, relieved, proud, satisfied, and a sense of self achievement.

Based on the results, it is always beneficial to recognize, respect, and celebrate each others' accomplishments that contribute to overall team morale. A study (Besker et al., 2020) also

supports this assertion, stating that showing appreciation and acknowledgment effectively encourages developers to repay TD since it fosters a sense of satisfaction. So, in general, if developers are praised, commended, or rewarded for paying technical debt, they gain a sense of self-worth, pride, and self-satisfaction. A correlation between a sense of appreciation and a sense of support presented in (Besker et al., 2020) indicates that developers who believe their effort to repay technical debt is recognized feel motivated to repay TD satisfactorily.

However, according to a respondent, praise is somehow like eating a chocolate bar that gives fun and energy only for that instant. Likewise, praises are also a short-term morale booster. In the long term, developers should believe in their responsibilities and quality of work rather than the praise. It is because paying technical debt should be considered as a regular part of the software development process.

The impacts of praise on developers' productivity

Appreciation or praise inspires developers to repay technical debt (Besker et al., 2020). If developers could repay the technical debt on time and within resources, it obviously helps to achieve higher productivity. The results show that praise boosts developers' morale both short and long term, so the developers with high morale, enthusiasm, and encouraged mentality can perform better and at a higher speed, enhancing overall productivity.

Praise could be a source of joy within the team, which often helps to increase developers' working speed and productivity. In addition, the results indicate that praise also helps to achieve quality because developers feel more responsible if they get recognized or appreciated. Achieving good quality in work is the crucial aspect of increasing productivity. So, usually, praise or appreciation/recognition for paying technical debts boost developers' morale and productivity.

6.2.4 The impact of team collaboration for repaying technical debt

Software engineering projects are fundamentally collaborative in nature, requiring many software engineers to coordinate their work to create a huge software system. Establishing a shared understanding between team members is imperative to develop the prominent software (Whitehead, 2007). Effective collaboration is arguably the most critical factor in attaining high-quality, efficient, and successful software engineering methods and outcomes in practically any software company (Bosch & Bosch-Sijtsema, 2010). Likewise, the results from this study indicate that team collaboration for repaying technical debt is constructive

and effective in increasing developers' morale and productivity. As I have mentioned earlier, the software development process is teamwork. Collaboration among the team members is crucial in developing quality software with predetermined resources and deadlines. Not only that, team collaboration makes it easier to refactor technical debt. Good coordination and high team morale can be achieved if a colleague helps other colleagues to repay or fix the TD.

Impact on developers' morale

The results show that team collaboration always encourages and satisfies developers. If a team member assists in repaying the technical debt, that makes any developer feel that they are also an inseparable part of that team, their existence matters. So, sort of these feelings inspires developers, and they approach their work with a greater sense of responsibility, which ultimately boosts their morale. This statement supplements (Hardy, 2010), which argues that an excellent personal relationship helps in attaining higher morale.

In software engineering, team collaboration is the most critical aspect of a successful software development process. (Codabux & Williams, 2013) argue that the collaboration introduces a sharing culture across the team and helps identify the person who needs help. Likewise (Codabux & Williams, 2013), results imply that team collaboration between developers for repaying technical debts brings them closer, provides an opportunity to know each other to make a strong team bonding. (Ghanbari et al., 2017) also explains that management of technical debt should be a collaborative effort in which all team members contribute to repaying the debts. Team collaboration is a key to making a fully encouraged, motivated, bonded, or concrete software development team. So, having good communication and a pair of extra eyes to identify and repay the technical debt boosts a developer and team morale.

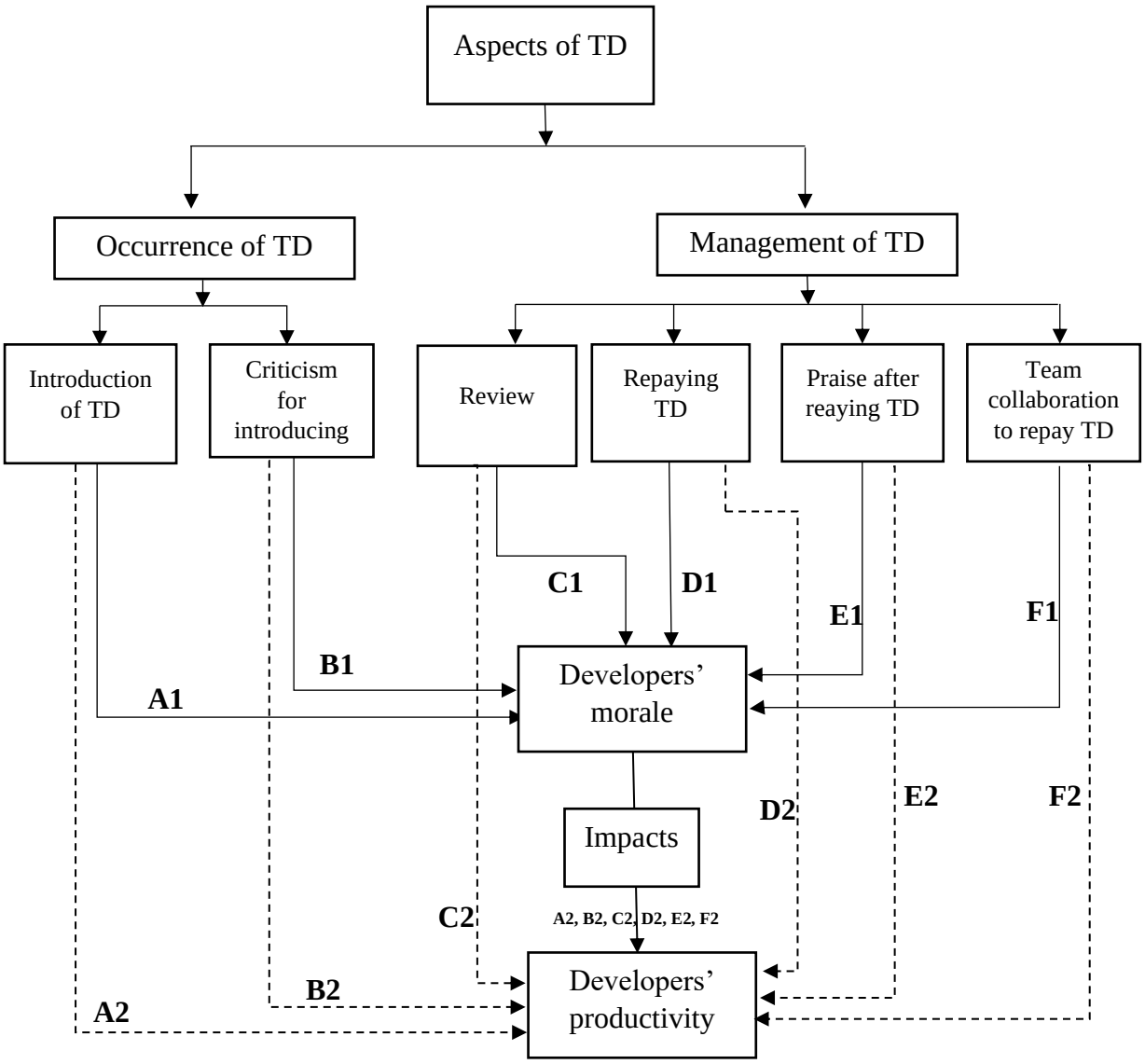
On the contrary, if there is no collaboration to fix the technical debt, division and grouping within the team may emerge. The results indicate that not having proper guidelines or help even though team members know about that is frustrating to developers. So, if there is no team collaboration, no developers feel good; instead, they feel an extra mental or psychological burden, and finally, their morale suffers.

Impacts on the productivity

The use of daily stand-up meetings, burndown charts, and good communication reduces the TD at a lower cost (Shriver, 2010), (Besker et al., 2020). The results also indicate that team collaboration minimizes the cost of all the mental pressure. Good communication and collective effort from the team to fix technical debt are vital for improving software quality and productivity. Managing technical debt collectively in a team boosts productivity (Ghanbari et al., 2017). In a team, there might be a situation where a particular developer could not identify and refactor technical debt. In such a situation, team collaboration and a pair of extra hands are essential to speed up the refactoring process, ultimately attain a higher productivity rate.

The working environment has a significant impact on the software developers' productivity. A team collaboration distributes positivity and good vibes among each other in the workplace that boosts both developers' morale and productivity. So, team collaboration unquestionably always increases developers' productivity by helping to refactor technical debt on time.

Relationship between different aspects of TD, developers' morale, and productivity



Arrow	Relation	Based on the results of	Arrow	Relation	Based on the results of
A1	Substantial negative impacts with few positive effects	5.1.1	**A2**	Usually lowers; however, few can perform better	5.1.2
B1	Positively impacts if given nicely, otherwise negatively impacts	5.1.3	**B2**	In general, it improves, however, few reduces the working speed	5.1.4
C1	Except in a few cases, positively impact	5.2.1	**C2**	helps in improving	5.2.2
D1	Negatively impacts while repaying, positively impacts after repaying	5.2.3	**D2**	Negatively impacts while repaying and boosts after repaying	5.2.4
E1	Positively impacts	5.2.5	**E2**	Positively impacts	5.2.6
F1	Positively impacts	5.2.7	**F2**	Positively impacts	5.2.8

Figure 13: Relationship between different aspects of TD, developers morale, and productivity

6.3 Implication

The thesis contributes to the human aspects of technical debt that specifically address the impact of technical debt on developers' morale and productivity. Different but essential aspects of technical debt such as the introduction of TD, review, paying TD, criticism and praise because of the technical debt, and collaboration to fix the technical debt have been explored independently and in-depth than other previous research. This thesis has few novel contributions and, at the same time, amplifies previous outcomes. Prior studies mostly examine the impact of technical debt on morale and productivity separately. However, this thesis collectively analyzes the impacts of technical debt on morale and productivity. It means the impact of developer morale (because of incurred technical debt) on developers' productivity has also been studied separately, which could be a novel outcome and research opportunity in this research area. The conceptual framework presented in this research explores the technical debt from the human perspective (mostly morale and productivity). It shows that different human-related aspects of technical debt (occurrence and management) such as review, criticism, praise, collaboration play an important role in developers' morale and productivity. Both researchers and practitioners can implement figure 13 above, which portrays how individual aspects related to the occurrence and management of TD affect developers' morale and productivity. So, overall, this thesis contributes to both academic and practitioners communities, which are explained as follows.

Implication for researchers

There are very few studies (Besker et al., 2020), (Ghanbari et al., 2017) that primarily investigate the influence of TD on developers' morale. However, it still lacks the proper investigation on impacts on developers' productivity due to the changed morale. This study strengthens and adds some novel outcomes in this research area. This becomes even more interesting considering the interviewees' controversial viewpoints about the review process. Even though the review process sometimes makes developers nervous, their productivity increases because of the review. This is not an ordinary finding because nervousness should decrease productivity, but it did not happen so. Thus, this research provides some bases to investigate the different relations of morale and productivity related to human aspects of technical debt.

Throughout the study, I have identified that the impact of technical debt on developers' morale and productivity can also be studied based on the different aspects related to the

occurrence and management of TD. As a result, this thesis is able to show that developers usually feel unsatisfied after introducing technical debt. However, it can fascinate developers in the short term. In general, after introducing technical debt, developers usually know that they must repay those debts at some stages of software development, due to which they can not be happy for a long and morale goes down.

Nonetheless, as a new outcome and potential further research opportunities, the results find that morale after introducing technical debt is subjective. It all depends on how technical debts perform. If TD works well as expected and is not liable to be repaid in the future, it significantly enhances developers' morale even in the long run, ultimately increasing productivity.

For researchers, this study facilitates the insights that criticism for introducing technical debt and praise for repaying technical debt have huge impacts on developers' morale and productivity. In general, no developers like to be criticized for introducing TD. Straight and forward criticism always demotivates, while constructive criticism always helps, and it does not hurt the developers' morale. On the other side, praise is always helpful to strengthen developers' morale. Developers always feel happy and motivated if they are praised or appreciated for repaying TD.

This study suggests that the impact of paying technical debt on developers' morale should be studied in two stages: developers' feelings while repaying technical debt and after successfully repaying TD. Most of the time, developers' feeling during the process of paying TD is not good. Their level of motivation will be reduced, and morale will suffer. On the other side, developers usually feel satisfied and motivated after paying TD, which boosts morale.

Another less spoken aspect of technical debt that should be considered on top is team collaboration. Team collaboration is hugely beneficial in every way. It assists in establishing a perfect relationship between team members, which increases team morale, helps in refactoring debt more quickly with fewer resources, and increases productivity.

Finally, exploring the impacts of technical debt on developers' morale and productivity based on different human-related aspects of technical debt could be a novel research approach. This approach could help researchers to explore more human aspects related to technical debt in a broader sense.

Implications for practitioners

The findings of the study also represent a meaningful contribution to practitioners. There are many implications that IT practitioners can implement during the software development process.

- Software practitioners can use the finding of this thesis to understand the impact of the review. Reviews always help to point out the issues or short corners taken during the software development process. So, in other words, reviews help to maintain quality by identifying the flaws in the software development process, whether in the code or process. Since the results show majorities of developers have a positive feeling about the review process, software practitioners should set a culture of reviewing, such as code review. A pair of extra eyes and skillful observation emphasizes the developers' morale and productivity.
- Most of the time, technical debt hurts morale and reduces overall productivity. So, our research suggests developers to follow all the quality guidelines that prevent them from introducing technical debt. However, if developers can maintain the pressure of paying technical later, it might be fruitful to introduce technical debt for the short run. For example, to be able to deliver the project to the customer within the deadline. However, introducing technical debt is always risky, which can hurt later.
- Our research suggests that repaying technical debt is a time-consuming and frustrating effort during the software development process. It withstands the speed of the software development process. Thus, this thesis recommends that software practitioners should avoid introducing known technical debt if it is possible.
- This study shows that if a team praises a member for repaying technical debt or fixing issues, a particular member's and whole team morale will increase. So, praise emphasizes developers' and team morale, which in essence increases productivity. Thus, this study proposes that software practitioners who work in teams should express their gratitude or appreciation for one another's successes.
- Likewise, criticism for introducing TD can be considered as a factor responsible for developers' low morale. As a result, it is usually preferable to refrain from pointing out or criticizing colleagues or team members. However, if someone inspecting the quality of the work want to provide a comment, it should be private, constructive, and positive as feedback.

- Finally, but certainly not least, team collaboration is essential for repaying technical debt. Team collaboration always facilitates developers feeling, motivations, and morale. The extra effort on the part of colleagues certainly aids in refactoring TD and strengthening productivity. Thus, this study proposes that practitioners should be concerned with making an easily understandable and well communicable team.

6.4 Threats to validity

Validity

The validity of a study refers to the acceptability of the results. It indicates the extent to which the findings are genuine and uninfluenced by the researchers' subjective viewpoint. All steps of the research must address validity (Runeson & Höst, 2009). There are different ways to classify aspects of validity and threats to validity in the literature. However, (Runeson & Höst, 2009) explains four aspects of the validity, which are given as follows:

Construct validity

This validity component reflects how the operational measures analyzed accurately reflect the researcher's intent and what is researched in response to the research questions (Runeson & Höst, 2009). So, concerning construct validity, there is a concern about the research question that interviewees might feel difficulties answering how technical debt affects human aspects (morale). It is because both technical debt and morale have comprehensive feasibility. To mitigate this, instead of asking about the impact of TD on morale directly, I asked about the impact of different aspects of technical debt on developers' morale. This helps interviewees in understanding technical debt and morale from depth.

Another construct could be that researchers and interviewees can differently interpret the questions. In order to mitigate this construct, a sample interview was taken with an IT engineer, and a proper explanation of the purpose of the study to the interviewees has given, which also helps them to understand the interview questions. We recorded interviews in audiovisual format, which enabled us to transcribe interviews more precisely and avoid misinterpretation of the collected data (Besker et al., 2020). Since we have chosen qualitative data analysis method to interpret the relationship between technical debt, developers' morale and productivity, future research might be needed to provide more meaningful insights.

Internal validity

Internal validity is relevant for examining causal relationships. When a researcher examines whether one element influences another, there is a possibility that a third factor also affects the researched factor (Runeson & Höst, 2009). There is a threat that developers' morale could be affected by other factors rather than aspects discussed in this thesis, such as introducing TD, reviewing, paying TD, getting criticism or praise, and team collaboration. In order to mitigate this threat, questions to developers were asked precisely in the form of semi-structured interviews. So, interviewees understood the concept of technical debt and its impact on morale correctly and did not deviate from other factors. Throughout the interview, Interviewees were also asked about the specific cases that they have introduced technical debt and its' consequences so that the interviewee at that particular time only thinks about technical debt and its' impact on morale. However, it is likely not possible to ensure that developers' morale has not been affected by other factors. So, further study in a controlled experimental environment is needed.

External validity

The external component of validity concerns the extent to which the findings may be generalized and the extent to which the findings are of interest to others outside the researched case (Runeson & Höst, 2009)**.** Since the results in this study are based on interviews, the threat to the external validity could be the number and selection of interviewees. Since the number of interviewees is 11 and they are from Nepal and Norway only. So, even though the interviewees have enough experience and knowledge in software engineering and technical debt, their responses might not be representative of the entire developer population. To minimize the risk of external validity, I sought to identify appropriate interviewees for the interviews. To ensure that the respondents chosen are suitable for this study, I set the following criteria: interviewees must be working in a software company and have a basic idea of technical debt. In order to mitigate the risk of a lower number of interviewees, I have tried to include interviewees from 11 different companies. However, further research should be done, including interviewees from various companies around the broader geographical range.

Reliability

The reliability factor considers the amount to which the data and analysis are dependent on the specific researchers (Runeson & Höst, 2009). Since qualitative interviews collected data via video conferencing, there is a threat to the reliability of collected data. Another threat could be the selection of the data analysis approach: thematic analysis using NVivo. To minimize the risk of reliability, the questionnaire for the interview were prepared from the supervision of two supervisors on multiple discussion. I recorded the interviews, recording every detail and making them more accessible during the coding and analysis. The interviews were coded and edited numerous times on suggestions from supervisors.

6.5 Limitation

This thesis has some limitations related to qualitative case studies and specific to this study.

- **Consideration of aspects of occurrence and management of technical debt**
  To investigate the impact of occurrence and management of technical debt on developers' morale and productivity, I examined six different aspects of technical debt affecting developer morale and productivity. However, those aspects were determined only based on antecedents of morale, a few prior research, interviewee responses, and personal experience. So, there might be other important aspects of technical debt that belong to occurrence and management that affect the developers' morale and productivity. As a result, more investigation will be required.
- **Shortage of interview time**
  Interviews were typically twenty to twenty-five minutes in length due to interviewees' schedule constraints. I believe that if those were a little longer, it would be friendlier and beneficial to elicit additional information.
- **Lack of real-time face to face interviews**
  Due to the Covid-19 pandemic, I could not perform real-time face-to-face interviews with the interviewees. The interviewees' real-time expression, feelings, body language, working culture, and environment matter as it is a qualitative study looking for opinions, feelings, and morale. So, it would be nice to conduct interviews in their working place with real-time expressions and emotions.

7. Conclusion

This thesis aimed to examine the impact of technical debt on developers' morale and productivity. I have presented a related theoretical context for the concept of technical debt in general and from the human aspects. Likewise, research design, methods, context, and results were also presented. I conducted semi-structured qualitative interviews with eleven software practitioners from different software companies. Lastly, results were addressed in relation to the research questions. This chapter will conclude the comprehensive study and provide recommendations for future research.

The first research questions aimed to investigate how the occurrence and management of technical debt affect the developers' morale and productivity. The impact of technical debt was studied based on the six different aspects of technical debt. Among them, I assume that 'introduction of TD' and 'getting criticism after introducing TD' belong to the occurrence of TD while the other fours aspects: 'review,' 'paying technical debt,' 'praise after repaying TD,' and 'team collaboration to repay the TD' belong to management of TD. Results indicate that most developers do not feel satisfied after introducing TD since they know that they are liable to pay those debts at some stage of software development lifecycles. So, introducing technical debt negatively influences developers' morale. However, short-term morale or happiness can be achieved if the debts help them instantly.

Regarding the other four aspects related to the management of TD, the review process helps developers improve the quality of their work. Apart from a few developers who sometimes feel nervous about being reviewed, review usually boosts their morale. On the other hand, the process of paying technical debt is time-consuming and stressful to developers, while they feel satisfied and motivated after paying technical debts. Likewise, results show that the praise for fixing technical debt has a positive role; it clearly boosts developers' morale. Similarly, team collaboration is an important aspect to be considered while managing technical debt. It creates a positive vibe in the team, increasing the overall team morale and connections. In contrast, lack of proper team coordination is responsible for more psychological pressure and declining developers' morale.

The second research question aimed to identify and discuss how morale affects the developers' productivity. The morale we discussed for research question two is the affected developers' morale after introducing technical debt. I concluded that the occurrence of technical debt affects the developers' morale and then their productivity. Developers' morale

generally declines with the introduction of TD, limiting developers' productivity, while the review helps in achieving higher productivity, although it sometimes makes developers nervous.

Since the management of technical debt positively impacts developers' morale, it motivates developers to maintain the quality of their work. The qualitative work, without a doubt, plays a vital role in preventing the occurrence of technical debt. If there is no TD, it increases the developers' performance since they do not need to spend their time repaying technical debt. Results also imply that team collaboration increases the developers' productivity by repaying the debt within deadline and resources because good cooperation within a team could improve work efficiency. On the other hand, praise after repaying TD makes the developers feel self-worth which is a real capsule to boost their productivity. Developers are usually not fond of paying technical debt, and their morale goes down while repaying technical debts hinders their productivity. However, developers after repaying TD can perform better since repaying technical debt could be a learning process.

7.1 Future Work

Despite the outcomes of this study being insightful and inspiring, additional research can provide a more detailed understanding of the relationships between the occurrence and management of TD, developer morale, and productivity. So, it would be meaningful to follow up on this study in the future, and there could be other interesting topics to research further.

This study only investigates the few aspects of TD and their impact on developers' morale and productivity. So, it could be fruitful to see the impacts of TD by considering some other measures of TD, such as documentation, communication, and presentation.

The results indicate that few developers' morale and productivity are unaffected by the occurrence and management of technical debt. Thus, it could be exciting to conduct additional research to identify factors that prevent developers from being affected by technical debt, such as developers' experience could be one factor.

It would be beneficial to conduct more case studies on this topic to validate and verify the finding of this study. Further studies should include a larger sample size with different roles from the broader geographical regions having different work cultures. Depending on the origin of the companies, the work culture could be different, affecting the occurrence and management of TD and the influence of morale and productivity.

References:

Aldaeej, A. (2019). Towards Effective Technical Debt Decision Making in Software Startups. *ACM SIGSOFT Software Engineering Notes, 44*(3), 22-22.

Allman, E. (2012). Managing technical debt. *Communications of the ACM, 55*(5), 50-55.

Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., & Avgeriou, P. (2015). The financial aspect of managing technical debt: A systematic literature review. *Information and Software Technology, 64*, 52-73.

Avgeriou, P., Kruchten, P., Ozkaya, I., & Seaman, C. (2016). *Managing technical debt in software engineering (dagstuhl seminar 16162).* Paper presented at the Dagstuhl Reports.

Aydin, C., & Anderson, J. (2005). *Evaluating the organizational impact of healthcare information systems*: Springer.

Baskarada, S. (2014). Qualitative case study guidelines. *Baškarada, S.(2014). Qualitative case studies guidelines. The Qualitative Report, 19*(40), 1-25.

Becker, C., Walker, D., & McCord, C. (2017). *Intertemporal choice: decision making and time in software engineering.* Paper presented at the 2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE).

Besker, T., Ghanbari, H., Martini, A., & Bosch, J. (2020). The influence of Technical Debt on software developer morale. *Journal of Systems and Software, 167*, 110586.

Besker, T., Martini, A., & Bosch, J. (2019). Software developer productivity loss due to technical debt—a replication and extension study examining developers' development work. *Journal of Systems and Software, 156*, 41-61.

Bosch, J., & Bosch-Sijtsema, P. M. (2010). Softwares product lines, global development and ecosystems: Collaboration in software engineering. In *Collaborative Software Engineering* (pp. 77-92): Springer.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., . . . Ozkaya, I. (2010). *Managing technical debt in software-reliant systems.* Paper presented at the Proceedings of the FSE/SDP workshop on Future of software engineering research.

Bryman, A. (2007). The research question in social research: what is its role? *International journal of social research methodology, 10*(1), 5-20.

Cecchetti, S. G., Mohanty, M. S., & Zampolli, F. (2011). The real effects of debt.

Codabux, Z., & Williams, B. (2013). *Managing technical debt: An industrial case study.* Paper presented at the 2013 4th International Workshop on Managing Technical Debt (MTD).

Creswell, J. W. (1994). Research design. In: Thousand Oaks, CA: Sage.

Creswell, J. W. (2009). *Research design : qualitative, quantitative, and mixed methods approaches* (3rd ed. ed.). Los Angeles: SAGE.

Cunningham, W. (1992). The WyCash portfolio management system. *ACM SIGPLAN OOPS Messenger, 4*(2), 29-30.

Damian, D., & Chisan, J. (2006). An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. *IEEE Transactions on Software Engineering, 32*(7), 433-453.

Dictionary, C. (2021). Meaning of criticism in English. Retrieved from https://dictionary.cambridge.org/dictionary/english/criticism

Durrheim, K. (2006). Research design. *Research in practice: Applied methods for the social sciences, 2*, 33-59.

Evans Data Corp, C. F., Stripe research. (2018). *The Developer Coefcient Sofware engineering efciency and its $3 trillion impact on global GDP*. Retrieved from https://stripe.com/files/reports/the-developer-coefficient.pdf

Fagerholm, F., & Münch, J. (2012). *Developer experience: Concept and definition.* Paper presented at the 2012 international conference on software and system process (ICSSP).

Fernández-Sánchez, C., Garbajosa, J., & Yagüe, A. (2015). *A framework to aid in decision making for technical debt management.* Paper presented at the 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD).

Gerring, J. (2004). What is a case study and what is it good for? *American political science review, 98*(2), 341-354.

Ghanbari, H., Besker, T., Martini, A., & Bosch, J. (2017). *Looking for peace of mind? manage your (technical) debt: An exploratory field study.* Paper presented at the 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM).

Graziotin, D., Fagerholm, F., Wang, X., & Abrahamsson, P. (2017). *On the unhappiness of software developers.* Paper presented at the Proceedings of the 21st international conference on evaluation and assessment in software engineering.

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

Graziotin, D., Wang, X., & Abrahamsson, P. (2014). Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ, 2*, e289.

Guba, E. G. (1958). Morale and satisfaction: A study in past-future time perspective. *Administrative Science Quarterly*, 195-209.

Guion, R. M. (1958). Industrial morale (a symposium): I. The problem of terminology. *Personnel Psychology*, 62.

Hagerstownmarket.org. (2016). Teamwork And Collaboration. Retrieved from https://www.hagerstownmarket.org/teamwork-and-

collaboration.html?fbclid=IwAR0zP4Fs0T765cxquMyTvqz1HK1qKt6elpbN2Ll7caTpGDUkTLDYNS2_NmE

Hahn Fox, B., & Jennings, W. G. (2014). How to write a methodology and results section for empirical research. *Journal of Criminal Justice Education, 25*(2), 137-156.

Hardy, B. (2010). *Morale: definitions, dimensions and measurement.* University of Cambridge,

Henderson, J., & Lepper, M. R. (2002). The effects of praise on children's intrinsic motivation: A review and synthesis. *Psychological Bulletin, 128*(5), 774-795.

Heurich, M., & Vignali, C. (2015). Innovations and its Impact on the Performance of Acute Care Hospitals in Germany-An Investigation Containing Empirical Research and Software Development. *Economics & Sociology, 8*(4), 149.

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *Ieee software, 29*(6), 18-21.

Laribee, D. (2009). Code cleanup—using agile techniques to pay back technical debt. *Code Cleanup-Using Agile Techniques to Pay Back Technical Debt, 24*. Retrieved from https://msdn.microsoft.com/en-us/magazine/ee819135.aspx

Li, Z., Avgeriou, P., & Liang, P. (2015). A systematic mapping study on technical debt and its management. *Journal of Systems and Software, 101*, 193-220.

Lim, E., Taksande, N., & Seaman, C. (2012). A balancing act: What software practitioners have to say about technical debt. *Ieee software, 29*(6), 22-27.

Longhurst, R. (2003). Semi-structured interviews and focus groups. *Key methods in geography, 3*(2), 143-156.

Marshall, M. N. (1996a). The key informant technique. *Family practice, 13*, 92-97.

Marshall, M. N. (1996b). Sampling for qualitative research. *Family practice, 13*(6), 522-526.

Matheson, J. L. (2007). The voice transcription technique: Use of voice recognition software to transcribe digital interview data in qualitative research. *Qualitative Report, 12*(4), 547-560.

Mattick, K., Johnston, J., & de la Croix, A. (2018). How to… write a good research question. *The clinical teacher, 15*(2), 104-108.

McLellan, E., MacQueen, K. M., & Neidig, J. L. (2003). Beyond the Qualitative Interview: Data Preparation and Transcription. *Field Methods, 15*(1), 63-84. doi:10.1177/1525822x02239573

McLeod, L., & Doolin, B. (2012). Information systems development as situated socio-technical change: a process approach. *European Journal of Information Systems, 21*(2), 176-191.

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and organization, 17*(1), 2-26.

Olsson, J., Risfelt, E., Besker, T., Martini, A., & Torkar, R. (2020). Measuring affective states from technical debt: A psychoempirical software engineering experiment. *arXiv preprint arXiv:2009.10660*.

Ozkaya, I. (2019). The voice of the developer. *IEEE Computer Architecture Letters, 36*(05), 3-5.

Perry, D. E., Sim, S. E., & Easterbrook, S. (2006). *Case studies for software engineers*. Paper presented at the Proceedings of the 28th international conference on Software engineering, Shanghai, China. https://doi-org.ezproxy.uio.no/10.1145/1134285.1134497

Peters, L. (2014). *Technical debt: The ultimate antipattern-the biggest costs may be hidden, widespread, and long term.* Paper presented at the 2014 Sixth International Workshop on Managing Technical Debt, Victoria, BC, Canada.

Peterson, C., Park, N., & Sweeney, P. J. (2008). Group well-being: morale from a positive psychology perspective. *Applied Psychology, 57*, 19-36.

Rashid, Y., Rashid, A., Warraich, M. A., Sabir, S. S., & Waseem, A. (2019). Case Study Method: A Step-by-Step Guide for Business Researchers. *International Journal of Qualitative Methods, 18*, 1609406919862424. doi:10.1177/1609406919862424

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering, 14*(2), 131-164.

Shriver, R. (2010). Seven Strategies for Technical Debt.

Spínola, R. O., Vetrò, A., Zazworka, N., Seaman, C., & Shull, F. (2013). *Investigating technical debt folklore: Shedding some light on technical debt opinion.* Paper presented at the 2013 4th International Workshop on Managing Technical Debt (MTD).

Tom, E., Aurum, A., & Vidgen, R. (2013). An exploration of technical debt. *Journal of Systems and Software, 86*(6), 1498-1516.

Victoria, C., & Virginia, B. (2017). Thematic analysis. *The Journal of Positive Psychology, 12*(3), 297-298.

Whitehead, J. (2007). *Collaboration in software engineering: A roadmap.* Paper presented at the Future of Software Engineering (FOSE'07).

Williams, C. (2007). Research methods. *Journal of Business & Economics Research (JBER), 5*(3).

Winchester, C. L., & Salji, M. (2016). Writing a literature review. *Journal of Clinical Urology, 9*(5), 308-312.

Wohlin, C. (2014). *Guidelines for snowballing in systematic literature studies and a replication in software engineering.* Paper presented at the Proceedings of the 18th international conference on evaluation and assessment in software engineering.

Yli-Huumo, J., Maglyas, A., & Smolander, K. (2014). *The sources and approaches to management of technical debt: a case study of two product lines in a middle-size finnish software company.* Paper presented at the International Conference on Product-Focused Software Process Improvement.

Zeitz, G. (1983). Structural and individual determinants of organization morale and satisfaction. *Social Forces, 61*(4), 1088-1108.

Appendices

A. Interview guide

Introduction

- Introduce myself and brief them about the thesis.
- Appreciate the interviewees for participating in this study.
- Read the consent form to ensure their confidentiality and anonymity.
- Request for permission to record the interviews

General Question

- What is your highest education level?
- What is your major at your highest education level?
- What is your role at this company?
- How many years of experience do you have in this role?
- How many years of overall experience do you have in the software engineering field?

Occurrence and management of TD

Review

- Do you have to follow specific company rules or guidelines (such as software process, tools, framework) or coding standards?
- How do you feel when someone reviews your work?
- What do you think code reviews will hinder or boost your morale and productivity?

Introducing TD

- Sometimes, we avoid implementing all of the quality standards and use sub-optimal solutions by violating quality standards.
    - How do you compare your level of satisfaction before and after violating quality standards (Introducing TD)?
- What are the immediate positive and negative feelings of introducing technical debts on your morale and productivity?
- How does the change of morale that happens after introducing TD influence your performance?

Criticism

- How do you feel if you get penalized or criticized for violating quality standards (Introducing TD)?
    - When you get criticized, does it make any impact on your morale and productivity?
- How do you feel if your colleague has violated the quality standards and gets penalized or criticized?

Praise

- How do your colleagues react when you fix quality issues?
- Have you ever been praised or appreciated?
    - How does praise or appreciation influence your morale and performance?

Repaying TD

- In the future, you might need to pay the debts that you have accumulated in the past. How do you feel when you actually spend time on paying debts instead of the development process?
- How does spending time on fixing issues/debts affect your productivity?
    - How much are technical debts effects completing the project within time, cost, and resources?
- How do you feel when you have fixed debts?
    - Do you feel more satisfied/happy? Why?

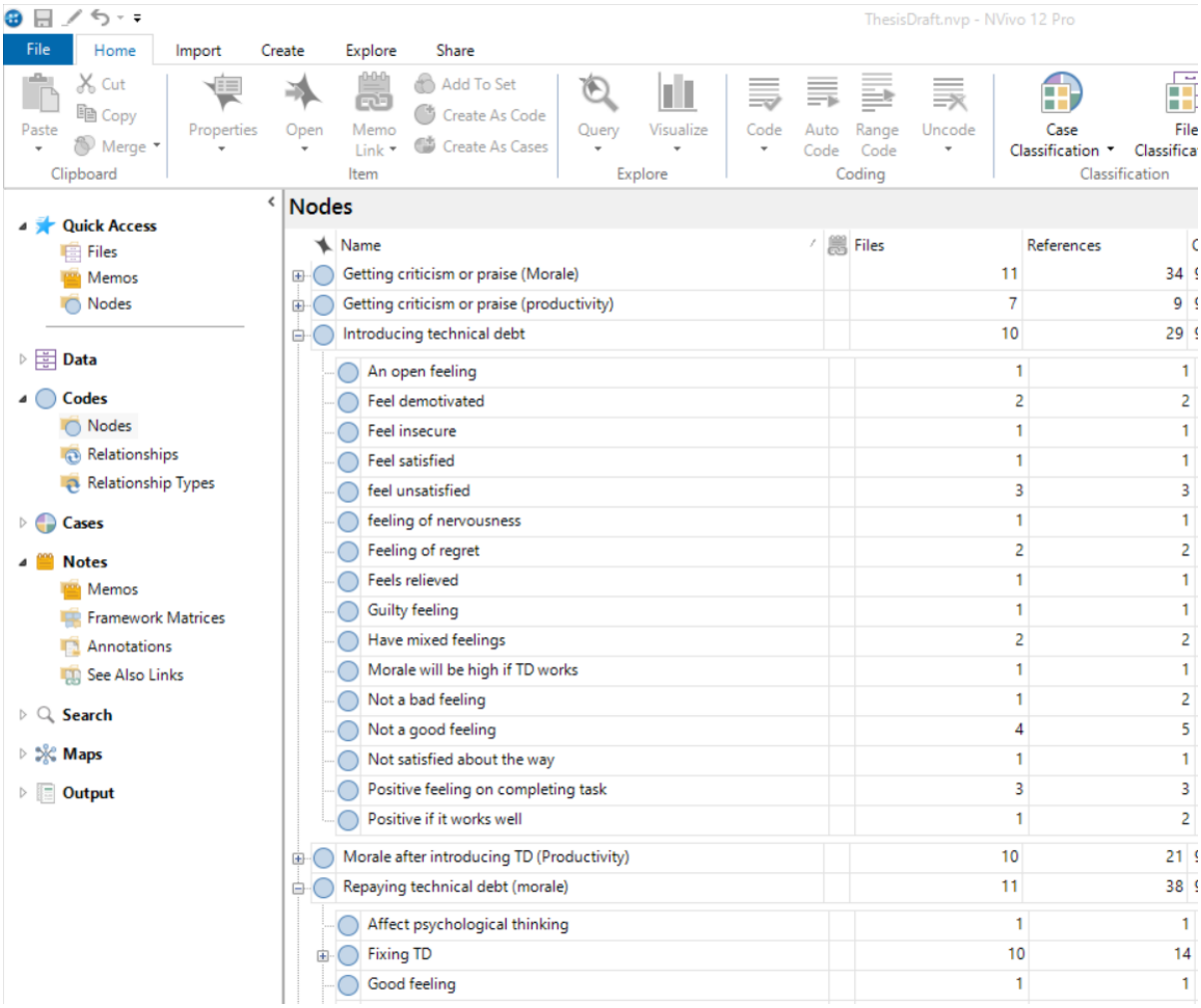
Team collaboration

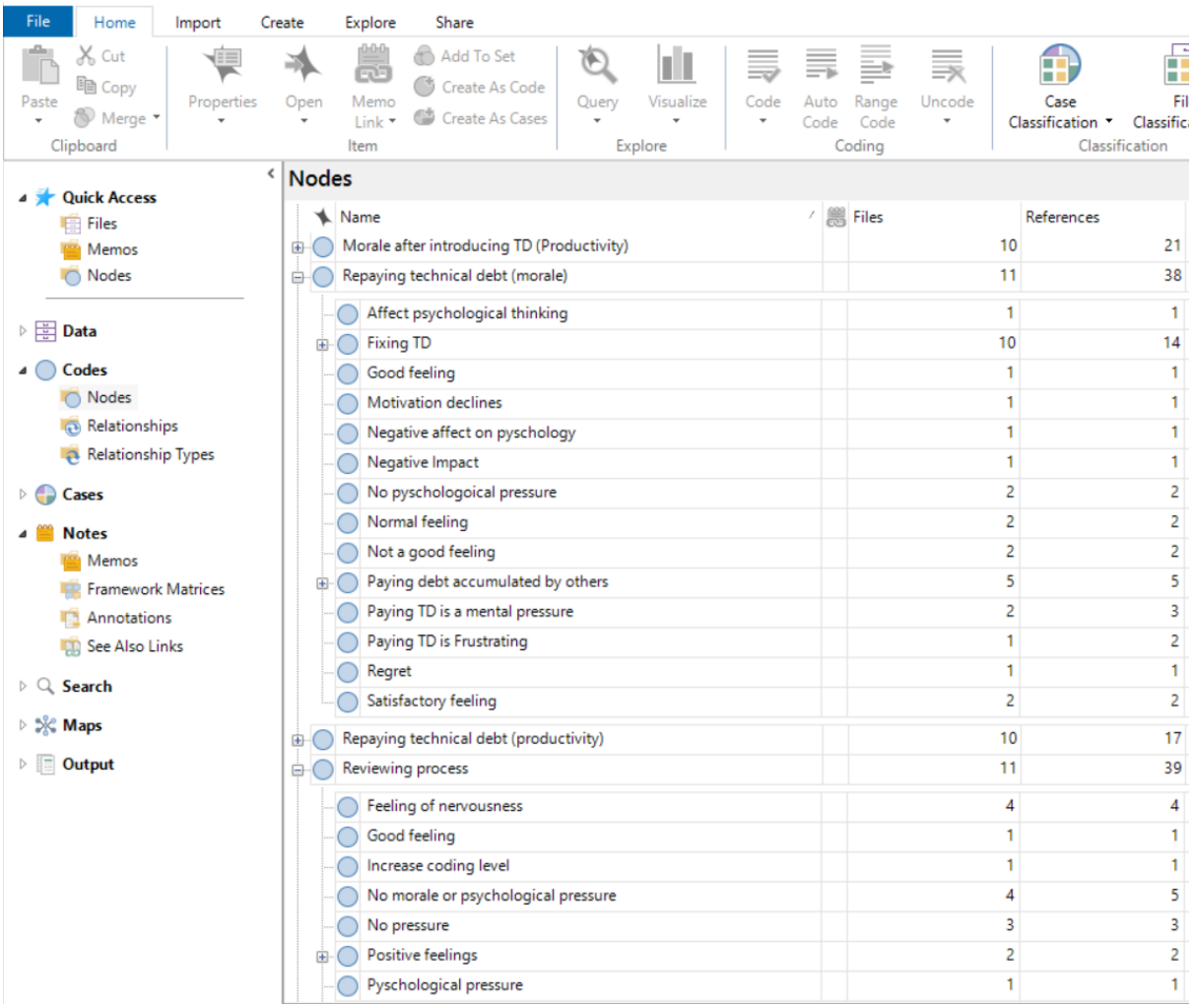
- How much do your colleagues help you with fixing the TD?
- How do you feel when your colleagues help you to fix the TD?
- How do you feel when they do not help or collaborate to manage the TD?

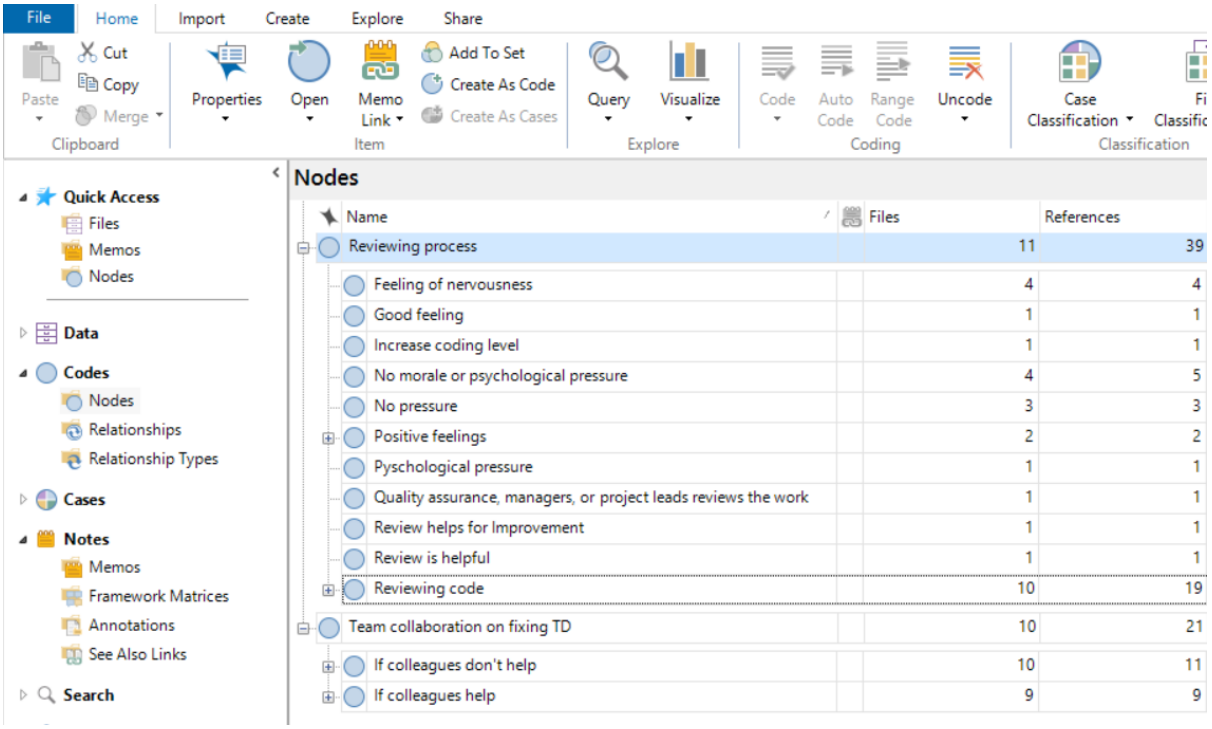
Closing

- Do you have any personal suggestions that you would like to add regarding TD and its' impact on your morale and productivity?
- Thank you for participating.

B. NVivo coding




File
Home
Import
Create
Explore
Share
Paste
Cut
Copy
Merge
Clipboard
Properties
Open
Memo Link
Add To Set
Create As Code
Create As Cases
Item
Query
Visualize
Explore
Code
Auto Code
Range Code
Uncode
Coding
Case Classification
Classification
Quick Access
Files
Memos
Nodes
Data
Codes
Nodes
Relationships
Relationship Types
Cases
Notes
Memos
Framework Matrices
Annotations
See Also Links
Search
Maps
Output
Nodes
Name
Files
References
Morale after introducing TD (Productivity) 10 21
Repaying technical debt (morale) 11 38
Affect psychological thinking 1 1
Fixing TD 10 14
Good feeling 1 1
Motivation declines 1 1
Negative affect on pyschology 1 1
Negative Impact 1 1
No pyschologoical pressure 2 2
Normal feeling 2 2
Not a good feeling 2 2
Paying debt accumulated by others 5 5
Paying TD is a mental pressure 2 3
Paying TD is Frustrating 1 2
Regret 1 1
Satisfactory feeling 2 2
Repaying technical debt (productivity) 10 17
Reviewing process 11 39
Feeling of nervousness 4 4
Good feeling 1 1
Increase coding level 1 1
No morale or psychological pressure 4 5
No pressure 3 3
Positive feelings 2 2
Pyschological pressure 1 1


File
Home
Import
Create
Explore
Share
Paste
Cut
Copy
Merge
Clipboard
Properties
Open
Memo Link
Add To Set
Create As Code
Create As Cases
Item
Query
Visualize
Explore
Code
Auto Code
Range Code
Uncode
Coding
Case Classification
Classification
Quick Access
Files
Memos
Nodes
Data
Codes
Nodes
Relationships
Relationship Types
Cases
Notes
Memos
Framework Matrices
Annotations
See Also Links
Search
Nodes
Name
Files
References
Reviewing process 11 39
Feeling of nervousness 4 4
Good feeling 1 1
Increase coding level 1 1
No morale or psychological pressure 4 5
No pressure 3 3
Positive feelings 2 2
Pyschological pressure 1 1
Quality assurance, managers, or project leads reviews the work 1 1
Review helps for Improvement 1 1
Review is helpful 1 1
Reviewing code 10 19
Team collaboration on fixing TD 10 21
If colleagues don't help 10 11
If colleagues help 9 9