Identifying and Managing the Impact of Team Communication and Team Organization on Technical Debt

A case study of Technical Debt

Deva Vaishnavi Rathnam
Divya Lal

Thesis submitted for the degree of
Master in Informatics
60 credits

Department of Informatics
[Faculty of Mathematics and Natural Sciences]

UNIVERSITY OF OSLO

[Spring 2021]

Identifying and Managing the Impact of Team Communication and Team Organization on Technical Debt

A case study of Technical Debt

Deva Vaishnavi Rathnam
Divya Lal

Abstract

Background: Technical Debt (TD) is a crucial concept in agile software development. TD arises in software systems because of various factors that need to be monitored and managed. This thesis is based on a case study relating to Team Communication and Team Organization, which are considered two major factors that influence the occurrence of TD in an agile team.
Aim: This thesis aims to study how software agile teams communicate when TD occurs and how the organization of teams in software companies affects the occurrence of TD. Communication of TD occurrence in agile teams is studied by analyzing types of communication, modes of communication and communication factors preferred by the teams. The team organization factors causing TD are analyzed along with few TD management techniques.
Method: A qualitative case study was conducted to investigate the occurrence of TD in various organizations. Data were collected by conducting 14 interviews in ten companies using semi-structured questions. The data was gathered, and interviews were conducted and recorded.
Results: The results show the various ways used to communicate the occurrence of TD and the types of communication used to discuss TD. Next, communication factors like team size and team diversity have both positive and negative effects on the occurrence of TD. Also, inter-team and intra-team communication have both effects on TD occurrence. Furthermore, direct communication of developers with customers and involvement in decision making has a positive impact on TD. The five main factors of team organization that cause TD mostly in agile teams are identified. They are employee turnover, inappropriate planning, lack of well-defined process, lack of knowledge and lack of training. In addition, TD activities discussed during planning, usage of common modules in an organization, selection of tasks by developers are identified as the ways of managing TD.
Conclusion: Choosing the right method of communication will help in reducing the occurrence of TD in a company. Small and co-located teams are preferred by the company to communicate which leads to less occurrence of TD. Also, most teams use daily stand-up and sprint planning meetings to discuss TD and their issues. The major factors causing TD due to the organization of teams are identified and certain features or aspects are analyzed, which help to reduce the occurrence and management of TD in organizations.

Acknowledgements

This thesis work has been done as part of our Master Graduate Program at the University of Oslo. The whole thesis is a shared work reflecting teamwork on all phases. The notion behind the thesis evolved as we were introduced to the course “Smart Process and Agile Methodology in Software Engineering” in our previous semester. This thesis would not have been to its final stage without the support and coordination of many people around us. Firstly, we express our deep gratitude to our supervisors, Antonio Martini and Yngve Lindsjørn, for their support, guidance, and patience without which this thesis would not have been possible. Furthermore, I am extremely grateful to all the participants of this study. Without their frankness, this study would not have been possible.

We are also extremely thankful to all our colleagues and participants who spent their valuable time and shared their experience to help us in gathering information and details about our topic. This acknowledgement would not be complete without thanking our parents and friends who supported us throughout our study and encouraged us to participate in the activities.

Deva Vaishnavi Rathnam

Divya Lal

May 2021

Table of Contents

List of figures

List of Tables

1 Introduction

The software companies perform their development process faster and more responsive, thus the time between analyzing the customer need and delivery of a solution is minimized. The current software development landscape involves the extensive usage of agile methodologies, where processes are defined, and practices are introduced to resolve the problems currently faced by the development teams (Martin, 2002)**.** One of the major advantages of agile software development is the quick release of the software functionality, which promotes the increase in deployment of Agile Software Development (ASD) (Dingsøyr et al., 2012).

However, the responsiveness in the short-term deliveries should not lead to less responsiveness in the long run. To illustrate such phenomenon, a concept called Technical Debt (TD) has been coined. ¨TD is a collection of design or implementation constructs that are useful in the short term, but set up a technical context that can make future changes more costly or impossible" (Avgeriou et al., 2016). However, the management of TD is often difficult in software projects since the presence of TD creates risks to the projects (Kruchten et al., 2012).

TD is an unavoidable phenomenon that exists in all projects and with all organizations (Lim et al., 2012). TD has both positive and negative effects on software projects (Li et al., 2015). The TD which is considered irrelevant in the initial stages of project development might lead to an increase in TD occurrence as the development phase continues. As this situation is certain to happen within the projects, necessary measures need to be taken to properly manage TD.

Even though the software engineering field has been investigating TD concepts over the past years (Rios et al., 2018), there exist only a few investigations on how agile team communication and organization have an impact on TD occurrence. To shed some light on this discussion, we found few papers related to impact of TD on team communication and team organization. The mapping of inter-team coordination mechanisms is performed by analyzing impersonal mode, group mode, and personal mode communication in a large-scale program (Nyrud & Stray, 2017). Inter-team coordination and intra-team coordination, which included team member turnover and team design choices are considered to know the impact on productivity (Melo et al., 2013). Some of the causes that lead to the occurrence of TD are investigated and the existing TD effects are studied (Rios et al., 2018). The best practices and challenges with agile adoption along with TD management techniques are analyzed (Codabux & Williams, 2013). However, the above studies lack focus on communication related to TD and discussed only a few factors that cause TD due to team organization. Also, the study does not provide a detailed description of their management techniques.

In this case study, we focused on the team communication factors, various modes of communication used by the software teams to communicate when TD occurs, and team organization factors that cause TD. Additionally, we also extended this study to identify some possible techniques for managing TD efficiently and its impact on the occurrence of TD.

The objective of this thesis is to bridge the gap by analyzing the way in which teams communicate when TD occurs, how team communication affects the occurrence of TD and the factors of team organization that affect the occurrence of TD. The questions to be answered in this research are:

R.Q 1: How do software teams communicate when TD occurs?

R.Q1.1: What are the types of communication used to discuss TD and its factors?

R.Q 2: How does team communication affect the occurrence of TD?

R.Q 2.1: How do communication factors impact TD occurrence?

R.Q 2.2: How does the direct communication of developers with customers have an impact on TD occurrence?

R.Q 2.3: How does the involvement of developers in decision making have any impact on TD occurrence?

R.Q 3: How does organizing a team in a software company affect the occurrence of TD?

R.Q 3.1: What are the team organization factors causing TD and how can they be managed?

R.Q 3.2: Does working with common modules have any impact on the occurrence of TD?

R.Q 3.3: Is there a need for a separate team and specific role for managing TD?

R.Q 3.4: What are the impacts of developers' task distribution on TD occurrence?

The first two research questions are related to team communication and the third is related to team organization. Answering the first research question helps the software teams to communicate without difficulties by choosing the right modes of communication and types of communication. The answer to the second research question helps the software teams to reduce the occurrence of TD by studying the comparative advantages and disadvantages to design a team. Also, teams can understand the importance of communicating with other software teams, collaboration with customers in person, and their involvement in decision making. Finally, answering the third research question helps to efficiently organize the software teams, which benefits the teams to effectively monitor and manage TD. To support this research question, the above-mentioned sub research questions are added, which helps the software teams to identify the significant factors that cause TD; factors that arise due to the inefficient organization of teams. Moreover, several TD management techniques such as usage of reusable components, assigning dedicated roles for managing TD activities, and work allocation of the developers help in understanding the effective ways of managing TD and reduce the occurrence of TD in agile teams.

1.1 Structure of the thesis

Chapter 2: Background gives a brief introduction to agile software development methodologies, team communication, team organization, and TD.

Chapter 3 Research Method describes the research method, research design, and data collection methods.

Chapter 4 Description of the studied Organization includes a detailed explanation of each organization studied.

Chapter 5 Results contain the results found in connection to topics which we discussed in Chapter 2.2 Team Communication and Chapter 2.3 Team Organization.

Chapter 6 Discussions presents the discussion of the results concerning the research questions.

Chapter 7 Conclusion and future work present the conclusion to the research questions and future work proposal.

2 Background

This chapter will present a background related to the thesis. Here we will introduce the relevant software development methodologies, as the methodology affects the various areas for communication and organization in a team. Second, background studies on Team Communication and Team Organization are presented. Lastly, a theoretical background of TD is presented.

2.1 Software development methodologies

2.1.1 Plan-driven development

Traditional software development, also known as plan-driven development, follows a sequential, systematic approach to software development. The features of these plan-driven approaches are: the software purpose and properties should be specified beforehand; a complete and elaborative project plan from the start of the project until the end needs to be prepared; detailed requirement specifications should be done and implementation of rigorous change request process is performed later; the architecture and design specifications need to be performed before the start of the implementation process; the programming phase focuses only on the programming tasks; the testing is performed at the end of the project, and a formal way of handling quality assurance is performed. The software models involved in the plan-driven development are the waterfall model, the Rational Unified Process (RUP), and the V-model (Petersen & Wohlin, 2010).

Some companies still use plan-driven models but in this case study, we focus on agile methodology.

2.1.2 Agile software development

A different development style showed its benefits over the traditional software development process since market forces, systems requirements, implementation technology, and project staff were transforming at a steadily increasing rate. The agile style of development directly addresses the problems of rapid change (Cockburn & Highsmith, 2001).

The values of agile software manifesto are (Highsmith & Cockburn, 2001):

- **individuals and interactions** over processes and tools,
- **working software** over comprehensive documentation,
- **customer collaboration** over contract negotiation,
- **responding to change** over following a plan.

Scrum

Scrum is a framework of agile methodology, which gives the flexibility to control and manage the requirements as well as the development of software (Hayat et al., 2019). It also contains sprints. A sprint is the smallest block of the scrum that has a small team that works on an assigned task. It usually lasts for 1 to 3 weeks (Srivastava et al., 2017). Each sprint aims to deliver a potentially organized and error-free product.

Scrum practices have the following roles, activities, and artefacts (Rubin, 2012):

- **Roles:** Product Owner, Scrum Master and Development team
- **Activities**: Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint retrospective, Product backlog grooming and Sprint execution.
- **Artefacts:** Sprint Backlog, Product Backlog and Burndown Chart.

Scrum Team and Roles

Scrum teams consist of three roles: scrum master, product owner and the development team (Rubin, 2012). They must be self-organizing and cross-functional without being dependent on others outside the team (Gonçalves, 2018).

Scrum Master

The scrum master has a leadership role over team members, and they help them to remove obstacles (Mahalakshmi & Sundararajan, 2013). The Scrum Masters’ main role is to eliminate obstacles that the team meets (Srivastava et al., 2017). They are responsible for protecting the team from outside interference and take a leadership role in removing obstacles that reduce team productivity (Rubin, 2012).

Product owner

The product owner maintains the product backlog (Mahalakshmi & Sundararajan, 2013). He is the one who is responsible for choosing which features to build and the order in which to build them (Rubin, 2012). The product owner represents the needs of all key stakeholders, the voice of the customer and the operational needs of the enterprise (Morris, 2017).

Development team

The development team is responsible for determining how to deliver what the product owner has asked for. The size of the development team ranges from five to nine people. The teams are self-organized to find the best way to accomplish the goal set by the product owner (Rubin, 2012).

Scrum Activities

Sprint

A sprint is the smallest block of scrum, consisting of a small team that works on the assigned task. The sprint usually lasts for 1 to 3 weeks (Srivastava et al., 2017). The changes include new requirements from the customer will not be accepted while in sprint (Mahalakshmi & Sundararajan, 2013). It is timeboxed so they always have a fixed start and end date (Rubin, 2012).

Sprint planning

The scrum team performs sprint planning to find the important subset of product backlog items to build in the next sprint (Rubin, 2012). It focuses on what to do and how to do it (Mahalakshmi & Sundararajan, 2013). The development team predicts what can realistically be developed during the sprint. The Product Owner discusses the objective to the scrum team about the work they want to achieve at the end of the sprint (Gonçalves, 2018).

Daily scrum

Scrum meetings can usually last for 15 minutes. The meetings involve a scrum master who is the chair of the team (Hayat et al., 2019). Three questions are generated in this sprint:

- What team members did yesterday?
- What team members plan to do till the next meeting?
- What are the difficulties faced by team members during the development process? (Mahalakshmi & Sundararajan, 2013).

It is also referred to as a daily stand-up meeting. It is conducted mainly to communicate the status of log items among the development team members (Rubin, 2012).

Sprint review

At the end of each sprint, there is a sprint review that takes place with the product owner to demonstrate the product that can be delivered (Srivastava et al., 2017). It focuses on reviewing the just completed features in the overall development effort (Rubin, 2012). It also helps them to motivate and strengthen the value of collaboration by giving a sense of achievement to the team's work (Morris, 2017).

Sprint retrospective

The teams discuss about what to be followed and not to be followed in order to improve further goes underway and the meeting will typically last for 15–30 minutes. It will be done after every sprint (Mahalakshmi & Sundararajan, 2013). During this activity, the scrum team will come together and discuss what is and is not working with scrum and related technical processes (Rubin, 2012).

Product backlog grooming

Refining, creating, estimating, and prioritizing the items collected in the product backlog is a grooming session (Rubin, 2012).

Artefacts

Sprint backlog

The collection of tasks along with their associated product backlog items is called sprint backlog (Rubin, 2012). The task for the sprint is decided by the sprint backlog (Hayat et al., 2019). It is a documentation of all the requirements for the current sprint to be worked on (Srivastava et al., 2017).

Product backlog

The product backlog is considered as the volume of requirement and is assessed by the product owner (Hayat et al., 2019). They are called user stories (Srivastava et al., 2017). The product owner is responsible for communicating it in the form of a prioritized list by finding and managing the sequence of work known as product backlog (Rubin, 2012).

Burndown charts

The sprint burndown chart is a displayed chart, which shows the remaining work in the sprint backlog (Rubin, 2012). It will be updated every day, and this gives a simple view of the sprint progress. It also provides quick pictures to team members for reference (Mahalakshmi & Sundararajan, 2013).

Kanban

Kanban is another project management methodology for software development that emphasizes "just-in-time" delivery. The focus of Kanban is to accurately state what work needs to be done, and when it needs to be done (Lei et al., 2017).

The following are the basic principles of Kanban for software development:

- Limiting Work in Process (WIP)
- Pulling value through the development process.
- Making the development process visible.
- Increasing throughput.
- Using a fixed backlog.
- Embedding quality

In Kanban, changes are allowed at any time and testing will be done after the implementation of each work product. One of the main workflow visualization tools used here is the Kanban board. It helps to optimize the work and guides the workflow by dividing the tasks into categories, including to-do works, in-progress works and works done (Matharu et al., 2015).

2.2 Team Communication

Communication is repeatedly mentioned to be one of the most important success factors in agile methodology. For a successful agile team, close collaboration and communication are essential (Sharp et al., 2009). Knowledge sharing between the teams is enhanced by communication and collaboration among software teams that may help to reduce the occurrence of TD (Codabux & Williams, 2013).

By analyzing the papers (Hummel et al., 2013), (Melo et al., 2013) and (Nyrud & Stray, 2017) we have considered factors such as team size and team distribution to know how communication is performed when TD occurs. Agile Software Development is mainly suited for small and co-located teams (Agerfalk et al., 2005). To prove this, factors such as team size and team distribution are considered (Hummel et al., 2013).

Team Size

Team size is defined as the number of team members on the software development project team. Teams having more than 25 members are more unwilling in taking responsibilities and scrum meetings take too long, which would take less time if team size would be of 5-10 results. This makes it difficult to communicate in large teams compared to small teams (Zia et al., 2018). An increase in team size is associated with the outcomes of TD (Besker et al., 2018). Communication in large teams is more difficult compared with small teams so the possibility of TD occurrence is more when the team size is large.

Team Distribution

Team distribution refers to the extent to which team members of SD projects are separated from each other (distributed) or close to each other (co-located) in terms of the physical location (Agerfalk et al., 2005).

- **Co-located teams**

Co-located teams depend on regular, synchronous, usually face-to-face interaction among all team members, including developers, customers, testers, and users. Effective communication is enhanced by making the participants present either virtually or physically at the same time. It includes face-to-face, telephonic conversation, and video conferencing (Shilpa & Ingle, 2010). Coordination theory was used to study two small co-located agile projects and found sprint planning meetings, open office space, and daily meetings provide efficient communication (Pikkarainen et al., 2008). To the best of our knowledge, we could not find any papers relating to TD and co-located teams.

- **Distributed team**

There are many benefits of using agile methods with distributed software development. It helps in evaluating and measuring the progress of the project and problems of the project are more easily noticed at the early stage. However, there are also challenges in using distributed agile software development like miscommunication in expressing TD related issues. Informal communication, which works well in co-located agile teams, is not possible in distributed teams. This also leads to a lack of trust amongst the team members. Thus, extra efforts are required on behalf of the team members to maintain effective communications (Shrivastava & Date, 2010). The distributed agile teams need to be aware and aligned to manage TD (Bavani, 2012). When distributed agile teams are aware and aligned, they can identify TD items easily and make optimal and informed Technical Debt Management (TDM) decisions.

Inter-team communication

Inter-team communication is communicating between the teams in terms of both co-located and distributed teams. In many organizations, software development teams often depend on other teams to complete their task (Ozer & Vogel, 2015). The software developer would receive more TD knowledge by communicating with other software developers, which helps to increase their performance. This will help to reduce the occurrence of TD.

Intra-team communication

Intra-team communication is communicating within a team in terms of both co-located and distributed teams. Team members need to collaborate effectively within the team but also with experts outside the team to accomplish their tasks and to reduce occurrences of TD. For example, designers, architects, infrastructure personnel, and other stakeholders (Dingsøyr & Brede, 2014).

Close collaboration with customers

Agile software development methodologies have realized the importance of communication in the successful development of a project. That is why agile software development emphasizes communication and close collaboration between development teams. The close interaction between developers and customer representatives usually leads to increased trust and a better understanding (Chau & Frank, 2004). One of the advantages is the requirements from the customer will not be missed if developers have direct contact with customers. Missing requirements might lead to requirement debt. Well defined requirements are considered as one of the preventive actions to avoid requirements debt (Freire et al., 2020). Another advantage is receiving early customer feedback. Getting customer feedback by collaboration can help clarify any requirements misinterpretations early. This early feedback will assist in preventing the wrong solution from being implemented, thus increasing TD (Codabux et al., 2014).

2.3 Team Organization

The organization of a team involves how well a team is formed to carry out the software development process. Agile software development supports self-organizing teams that organize themselves by choosing, committing, and carrying out their tasks (Hoda & Murugesan, 2016). The tutorials on agile planning help the teams to move forward with the software development process and a release plan of the product should be prepared using a given set of user stories (McDowell & Dourambeis, 2007). When the developers are not actively involved in the planning process, it results in weak team orientation (Stray et al., 2011). The requirement analysis from the users should be performed in a well-defined manner. The change in requirements needs to be accepted and validation of requirements must be accomplished. The Software Process Improvement (SPI) considers that a well-managed organization with a defined engineering process can develop software that meets the customer requirements within the estimated budget and schedule (Fritzsche & Keil, 2007). Employee turnover causes negative impacts on the development process, which leads to a decrease in overall organizational performance (Dalton & Todor, 1979). Organizational training helps the team members to generate an idea about teamwork, understand new tools to manage team-related issues, and improve the overall effectiveness and well-being of employees (Gren et al., 2019).

There are cases where TD needs to be dealt with when there are limited resources, certain factors should be combined which helps teams to analyze the critical nature of TD items regarding software future release (Codabux et al., 2017). When working agile, it seems in many cases that insufficient knowledge, lack of well-defined process, inappropriate planning, and deadlines are considered as some of the main causes for the increase in TD (Rios et al., 2018). Once the TDs are identified, it helps the development teams to take respective actions so that the occurrence of TD items can be prevented (Rios et al., 2018).

The main factors listed out to reduce TD occurrence (Freire et al., 2020) are:

- Project planning: There should be proper and well-defined planning that needs to be considered and followed, which describes actions related to project planning activities. This includes appropriate planning of projects, effective and efficient monitoring, deadline, and task allocation.
- Process definition: This refers to the well-defined process in the project and flexible changes in the process defined, upgrade and understanding the software development process.

Work or tasks allocated to the developers are also considered as a taxonomy of TD (Codabux & Williams, 2013). If the developer does not possess good knowledge of the tasks he is assigned, it may also become one of the causes for the occurrence of TD. The TDs thus occurring can be reduced by providing technical training to teams on design and code practices (Avgeriou et al., 2016). The developer's negligence to certain tasks, thinking to do them later, makes them go back and perform those, which leads to TD. To the best of our knowledge, we did not find any papers demonstrating the impact of using reusable components on TD, which is explored in this thesis.

Even though the focus is on developing new features, assigning dedicated teams to manage and be involved in TD activities helps in the reduction of TD occurrence (Codabux & Williams, 2013). When TD management is discussed during planning meetings, the capacity of the team performing TD reduction can be determined. Moreover, sprint planning should reserve 20% of their total time to discuss TD reduction. These actions may be considered as a best practice for effective TD management strategy and motivate several other companies to adopt the technique to manage TD (Codabux & Williams, 2013).

2.4 Technical debt

"TD is a collection of design or implementation constructs that are useful in the short term but set up a technical context that can make future changes more costly or impossible" (Avgeriou et al., 2016). Agile software development seems to have both positives and negatives in adding TD. Agile provides an iterative process that helps to iteratively keep track of TD. On the other hand, the Agile process and principles favor the focus on the features of the product and thus adds TD (Martini et al., 2015). Scrum agile methodology is often used by software developers in a project to develop a product in an efficient manner. Various factors of team communication and team organizations related to TD are discussed along with TD management techniques.

Some of the TD factors such as cost, decision making, risk, causes and time constraints are considered to know how teams communicate when these factors occur in the team. These factors are discussed below.

2.4.1 Cost

TD can also be defined as the future costs attributable to known structural flaws in production code that need to be fixed; a cost that includes both principal and interest (Bill et al., 2012).

The concepts of TD are explained by (Ampatzoglou et al., 2020):

- Principal - The effort required to eliminate inefficiencies in the current design or implementation of a software system.
- Interest amount - The additional development effort required to modify the software (adding new features or fixing bugs).

The cost of managing TD involves many different activities (Guo et al., 2016):

- Identification of TD items
- Analysis and evaluation
- Communication
- Decision making whether to pay back or avoid TD
- Business cost
- Documentation

Identification refers to developers finding low-quality software in the module. Analysis and evaluation refer to understanding TD items, which includes calculating principal and interest.

Communication refers to meetings between the project leader and developers to get information about TD management (Rios et al., 2020).

2.4.2 Causes

Causes of TD are categorized into eight types; these are development issues, external factors, infrastructure, lack of knowledge, methodology, organization, people and planning & management (Rios et al., 2020).

- **Development issues:** Issues that occur during the development phase in the project like bad code, change in design and requirements, inaccurate requirements, and lack of quality are some of the issues in the development category.
- **External Factors:** Causes that are external to the development team and organization. Some of the causes of external factors include pressure, a third-party team involved in the project, and the customer not listening to the project team.
- **Infrastructure:** The causes related to tools, technologies, and development environments, such as inadequate use of tools and unavailable infrastructure.
- **Methodology:** Lack of refactoring, the documentation does not exist, lack of requirement analysis and outdated or incomplete documentation are some of the causes related to processes and methodologies used in the development of the project.
- **Organizational:** The causes related to an organization such as inadequate management decision, lack of qualified professionals and the company does not give importance to documentation.
- **People:** The causes directly related to members of software development teams, like lack of commitment, lack of interest in acquiring knowledge, lack of team communication, lack of commitment and overload of the team.
- **Planning and Management**: The causes related to project planning and management are cost, deadline, poor allocation of resources, Manager’s lack of awareness of customer’s needs, inadequate impact and risk analysis, and inaccurate time estimate.

2.4.3 Decision Making

Deciding whether to pay back or keep the TD as such is one of the important factors to be considered while managing TD. An organization must choose to leave TD as such if the value of changing the system is less than the cost of those changes (Brown et al., 2010). But, if an organization decides to pay back the TD, it is often impossible to directly find the principal or the interest rate. Measuring principal and interest amount will be useful to make decisions on whether to keep or pay back TD.

Once decided to repay the TD, the one to repay first must be chosen because there can only be limited resources available for TD repayment. Therefore, prioritizing TD is the best way to use such resources to find which TD items must be paid first (Alfayez et al., 2020).

Many decision-making criteria can be considered while prioritizing. The criteria for prioritizing TD are grouped into four categories such as nature of TD, customer, effort, and project (Ribeiro et al., 2016) These categories will help the developers' team to decide at which time to pay off a TD item. The nature of the TD criteria is considering properties of TD such as seriousness and time of the TD incurred. Customer criteria is regarding any impact that TD has on customers. An effort criterion is related to the cost of TD that is the impact of TD on the project effort needed to pay the TD item. Project criteria is a lifetime and the possibility of evolution of the project.

2.4.4 Risk

Every project carries some amount of risk (Martini & Bosch, 2015). TD emerges from the poor structural quality and affects business both as to cost and business risk (Bill et al., 2012). The level of risk that an individual, team or organizational level is prepared to accept can influence decisions related to TD in different ways. TD can influence risk positively or negatively in the short term (Tom et al., 2013). Having a lower level of risk can influence decisions that create TD to reduce a project's delivery risk in the short term and having a higher level of risk can influence decisions to create TD although it may be increasing project risk.

If a team is approaching a project from a risk management perspective, paying off TD before making any changes is the more responsible move.

2.4.5 Time Constraints

Time constraints are one of the important factors that cause TD. The lack of time is the most common cause identified for the development. Lack of time also creates a lot of pressure on the development team, which leads to the occurrence of TD. Complex source code leads to the use of glue code to make it easier and cheaper for a time being rather than fixing the bigger problem. (Yli-Hummo et al., 2014)

2.4.6 TD Management

When the development and delivery of the projects become quicker without spending much time on design and longer-term, it results in huge TD. When new features are added to the existing TD, the complexity of the development process also increases and thereby results in a decrease in the overall quality and productivity (Yli-Huumo et al., 2015a). Therefore, TD needs to be managed and controlled thereby avoiding more accumulation to the existing TD. For this, firstly identify the TD and factors responsible for it. Then, during planning, the identified TD related tasks need to be put in the backlog together with all other items (Kruchten et al., 2012). TD can be managed only if there is a good understanding of the state of the art of TDM (Li et al., 2015). The TDM activities here include:

- **TD Identification** spot out TD occurred due to intentional or unintentional decisions and identify the factors causing TD.
- **TD Prioritization** orders TD to standard rules thereby picking up TDs according to their rank assigned starting with the high priority ones first.
- **TD Prevention** focuses on avoiding TDs from occurring.
- **TD Monitoring** keeps track of the cost and benefit of TDs that remains unresolved.

The best way to resolve TD is to include discussions on it during planning sessions (Power, 2013). The researchers investigate two examples of product teams located in a large company: One team, which is actively involved in a reduction of TD and another team that reflects negligence in the investment of TD reduction. The former team included TDM in their planning meetings and thereby determining the capacity of a team undergoing TD reduction and discussions about TD repayments. The latter team avoided TD management during their planning session but gave priority to new features and to fix defects, but they did not realize that most of the defects are because of TD, which later causes the system to destabilize. So, it is better to consider managing TD early during a planning session. This helps to make aware every team member about the progress and status of TD. Another way of managing TD is that the developers' tasks need to be analyzed with code reviews before they are out for the release, which helps to identify TD and is therefore able to resolve them (Mäntylä & Lassenius, 2008).

Documentation also plays an important role in managing TD occurrence. Developers will be aware of the existing TD if it is documented (Rolland & Lyytinen, 2021). If a system has a poorly designed module, then leaving documentation without updating makes the situation worse when the module needs to be modified, which leads to more occurrence of TD (Guo & Seaman, 2011). Also, if documentation is inadequate, developers might have to spend more time managing and working with TD issues (Tom et al., 2013).

3 Research Method

This chapter will summarize the method used in this study and the reason for choosing the method and the details about research design. First, research design is presented. Then preliminary study and type of research used is presented. Next, the details of data collection are presented. Lastly, data analysis is presented.

3.1 Research Design

The research design presented in Figure 1 shows the phases of data collection (black boxes), data analysis (white boxes), methods used to collect data (rectangle box), results (white ellipses) and an arrow that represents the flow of information from activity to result.

We conducted a preliminary study in phase 1, followed by qualitative data collection through interviews in phase 2 to collect data for our research. In phase 1, we found a research gap that there are only a few studies in TD occurrence related to team communication and team organization. Also, factors of team organization such as employee turnover in the team, inappropriate planning, lack of well-defined process, lack of knowledge, lack of training and factors of team communication such as team size and team distribution are found. The inter-team and intra-team communication are also found in the preliminary study. Along with, TD mitigation strategies are studied. In phase 2, we collected data considering the research gap as motivation and factors as a part of questions in the interview. Data is collected through semi-structured interview questions. The collected data is analyzed in phase 3 through thematic analysis. All the text in the interviews was transcribed and coded as TD occurrence related to team communication and team organization. And these are mapped into themes as the impact of TD occurrence on a team. A detailed description of phase 1 is explained in Chapter 3.2 Preliminary study, phase 2 is explained in Chapter 3.5 Data collection and phase 3 is explained in Chapter 3.6 Data analysis

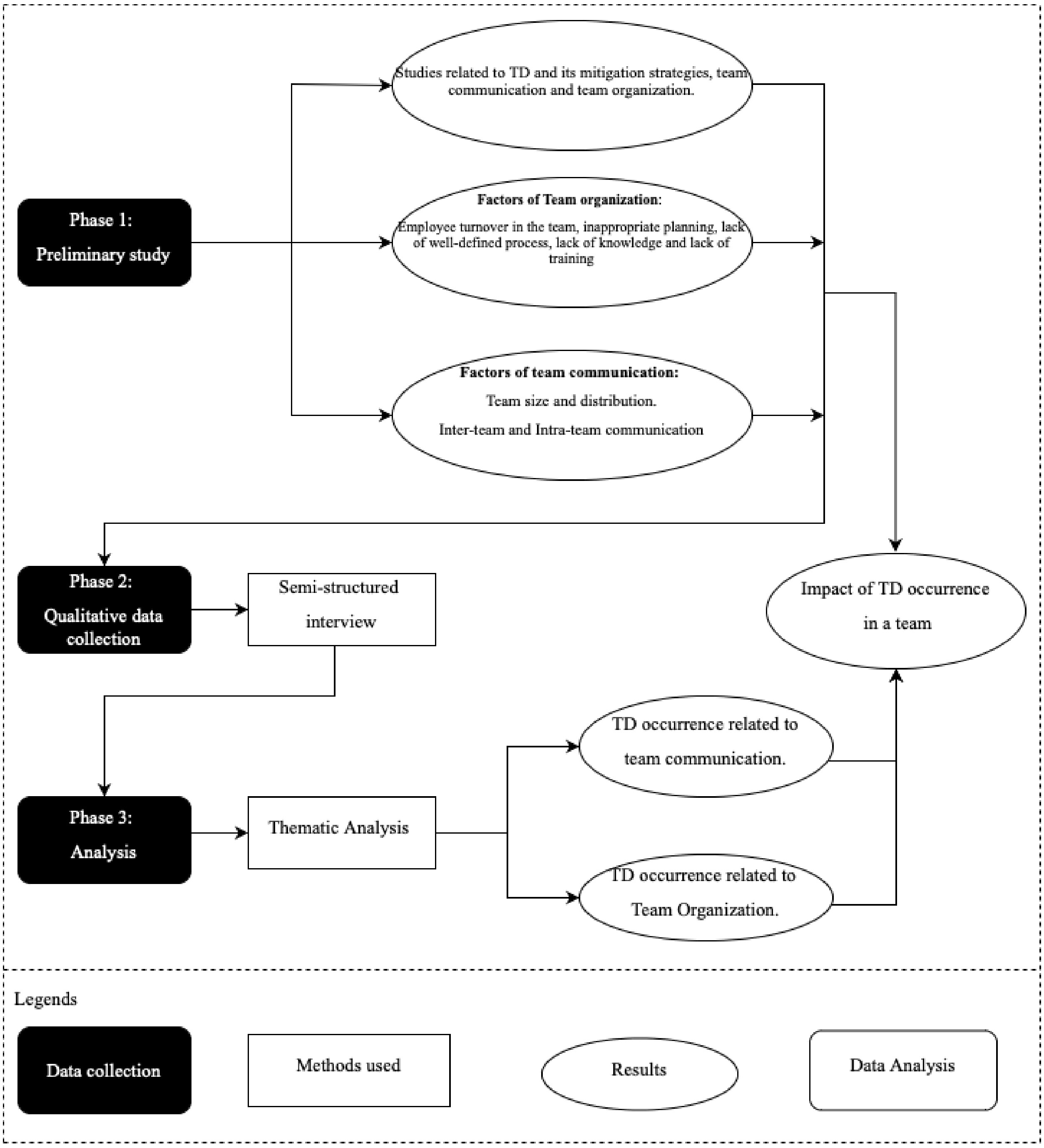

Studies related to TD and its mitigation strategies, team communication and team organization.
Phase 1: Preliminary study
Factors of Team organization: Employee turnover in the team, inappropriate planning, lack of well-defined process, lack of knowledge and lack of training
Factors of team communication: Team size and distribution. Inter-team and Intra-team communication
Phase 2: Qualitative data collection
Semi-structured interview
Impact of TD occurrence in a team
TD occurrence related to team communication.
Phase 3: Analysis
Thematic Analysis
TD occurrence related to Team Organization.
Legends
Data collection
Methods used
Results
Data Analysis

Figure 1: It represents the visualization of the research design and research method used in each phase. We show phases of data collection in black boxes, methods used in white rectangles, results in white ellipses, data analysis in white box and the arrow marks represent the flow of information.

3.2 Preliminary study

We studied several papers and found that two factors such as Team Communication and Team Organization are crucial to study as there are only a few papers on how team organization and team communication have an impact on TD; therefore, we wanted to concentrate our study on these factors.

Regarding team communication, we analyzed (Melo et al., 2013), (Hummel et al., 2013) and considered the communication factors, team size and team distribution to know how teams communicate when TD occurs. Also, we analyzed (Nyrud & Stray, 2017) and found it interesting to gain more knowledge about how communication is performed in both intra-team and inter-team cases. The author studied inter-team coordination mechanisms by analyzing three coordination categories: impersonal, personal and group mode. Personal mode is further divided into vertical communication and horizontal communication and group mode is divided into scheduled and unscheduled meetings. We have considered the coordination categories personal and group mode to find how TD is communicated. So, our first research question is based on the above analysis. We have also considered TD factors like risk, cost, time constraints and causes to know how these factors are communicated by members of a team. Finally, we studied (Dingsøyr et al., 2012) and considered customers' involvement with developers and including developers in decision making to know whether it has any impact on TD. The second research question is related to the impact of TD occurrence.

Regarding team organization, we analyzed the primary factors that cause TD that results from organizing teams in an organization (Rios et al., 2018), (Codabux & Williams, 2013) such as inappropriate planning, lack of well-defined process, lack of knowledge and lack of training. Additionally, we investigated whether there is a need for a dedicated team (Codabux & Williams, 2013) and specific roles within teams particularly to manage TD activities. Moreover, we analyzed and found the importance of the usage of common modules or reusable components by organizations and their impact on TD. Finally, we studied the developers' task distribution and task selection. Also, valuable information about their impact on TD are analyzed.

3.3 Case study

The case study can be defined as “an intensive study of a single unit to generalize across a larger set of units” (Gerring, 2004). It is conducted to investigate contemporary phenomena in their natural context, allowing the researcher to understand how the phenomena interact with the context (Runeson et al., 2012). The study helps to describe, compare, evaluate, and understand different aspects of a research problem. The five criteria for a case study are (Perry et al., 2004):

1. The research questions are formulated at the beginning of the study.
2. Data is collected in a planned and consistent manner.
3. Inferences are made from the data to answer the research question.
4. Explores a phenomenon, or produces an explanation, description, or causal analysis of it.
5. Threats to validity are addressed systematically.

We performed a case study involving ten companies to gain in-depth knowledge and to investigate the occurrence of TD in various organizations concerning team communication and team organization. Data collection through interviews is important in case studies. Interviews can be divided into unstructured, semi-structured and fully structured interviews (Runeson et al., 2012).

- **Unstructured interviews**: The interview questions are formulated as general concerns and interests of the researcher.
- **Fully structured interviews**: All questions are planned, and all questions are asked in the same order as in the plan.
- **Semi-structured interviews**: Questions are planned, but they are not necessarily asked in the same order as they are listed. Additionally, semi-structured interviews allow for improvisation and exploration of the studied objects. For these reasons, we have chosen to conduct semi-structured interviews.

The reason for choosing these ten companies was to gain more knowledge about the occurrence of TD in agile software development and to compare how TD was communicated in different organizations. Data were gathered by conducting 14 interviews over one month.

3.4 Qualitative research

There are three types of research designs: qualitative, quantitative, and mixed methods. The selection of a research design is also based on the nature of the research problem or issue being addressed, personal experiences of the researcher, and the audiences for the study (Creswell, 2009). Research designs are types of inquiry within a qualitative, quantitative, and mixed methods approach that provides specific direction for procedures in a research study (Creswell & Creswell, 2018).

Qualitative research is characterized by understanding some aspect of social life, and its methods, which generate words, rather than numbers, as data for analysis (Bricki & Green, 2007). The process of research involves emerging questions and procedures, data collected in the participant's setting, data analysis by analyzing the interview answers and changing into general themes, and the researcher making interpretations of the meaning of the data (Creswell, 2009). Quantitative research is a means for testing objective theories by identifying the connection among variables. These variables, in turn, can be measured, typically on instruments, so that numbered data can be analyzed using statistical procedures (Creswell, 2009). Mixed methods research involves combining or integration of qualitative and quantitative research and data in a research study (Creswell & Creswell, 2018).

For this study, a qualitative research approach was selected. As the research questions of this study involved examining how individuals interact with one another in their natural setting, a qualitative research approach seems to be suitable for this study.

To reduce bias by individual researchers, the interview is being conducted by multiple researchers. Also, analysis and findings are performed by multiple researchers. This helps to increase the validity of the study.

3.5 Data collection

The results and conclusions presented in this thesis are based upon data collected through semi-structured interviews, notes taking, and recordings. But it still ensures flexibility, it allows the interviewee to bring up ideas and questions based on the answers of the interviewee. During the interview sessions, it is recommended to record the discussion in a suitable audio or video format. Even if notes are taken, in many cases it is difficult to record all details, and it is difficult to know what is important to record during the interview. Possibly, a dedicated and trained scribe may capture sufficient detail in real-time, but the recording should at least be done as a backup (Runeson et al., 2012). Data collection should be based on a goal-oriented measurement technique, such as the Goal Question Metric method (GQM) (Basili & Weiss, 1984). In GQM, goals are first formulated, and the questions are refined based on these goals, and lastly, metrics are derived based on the questions. Following this method helps to collect quality data and the collection of unnecessary data can be avoided.

The main goal of the interview is to attain details of how teams communicate and coordinate tasks when TD occurs, to identify the team organization factors causing TD, the roles responsible for handling TD activities within the organization and to gain knowledge about different ways to manage TD. The sample population was selected using a non-probability sampling technique, where the interviewees are selected based on purposive sampling. Purposive sampling is selecting a participant who is most useful to the purposes of the research. Our main criteria for selecting interviewees were as follow:

1. The informant should be working in agile development.
2. The informant should be aware of TD concepts.
3. The informant must possess a minimum of two years of development experience.

We contacted people by sending them emails. The data collection consists of 14 interviews from ten different companies of different roles. The company and interviewees details are described more in Chapter 4. The interviews varied from 40 to 75 minutes. They were recorded with the consent of the people interviewed. This was to provide a precise interpretation of the interview. The data was collected in a one-month period where interviews are conducted via Zoom. The interviews conducted were semi-structured, which had topics related to team communication, team organization, and TD in the company. This also helped to provide TD related issues experienced in various organizations and discussed ways to manage this TD. An interview guide was used while interviewing to keep the direction of the interview. All interviews were transcribed, and then analyzed using NVivo[1]. Data Analysis is briefly described below in the Chapter 3.6 Data analysis

[1] NVivo is an analysis software developed by QSR International, www.qsrinternational.com

3.6 Data analysis

The thematic analysis looks across all the data to identify the common issues that recur and identify the main themes that summarize all the views collected by the interviewer (Bricki & Green, 2007). We used thematic analysis to identify, analyze patterns and themes with the data from interviews. The recorded qualitative data from interviews were transcribed. All the data sources were uploaded into a program called NVivo, which is a qualitative data analysis software. The first step is coding the data that is part of the interview; texts are given a code representing a theme. Next, codes are formed as sub-codes. The final step is to set a theme by arranging the different codes and sub-codes into themes. Figure 2 shows some examples of interview answers that are coded and mapped into themes. The complete interview answers coded in NVivo are mentioned in Appendices B. For example, one of the interview answers is ¨*I prefer a co-located team to communicate TD issues because it is much easier to connect with people.* ¨ It is coded as team distribution as this explains that team distribution plays a major role in communicating TD. And then it is mapped as team communication as it describes more about communicating TD.

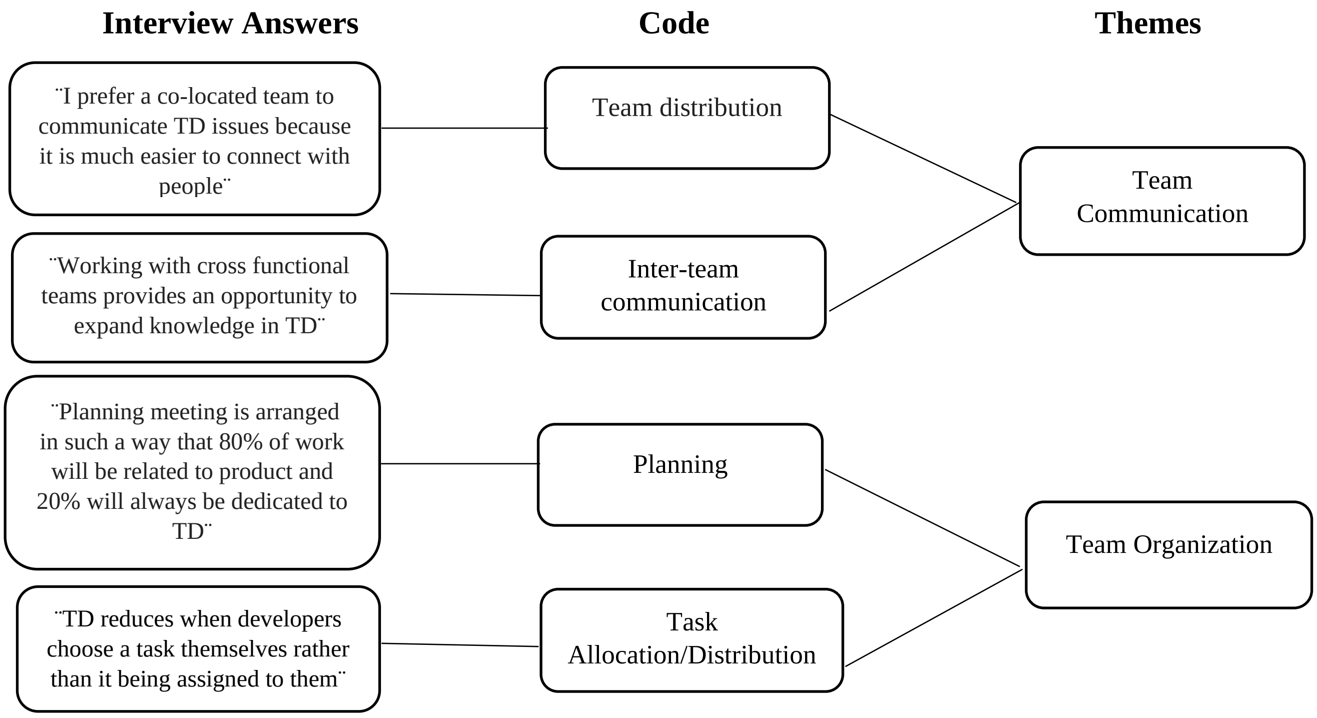

Interview Answers
Code
Themes
¨I prefer a co-located team to communicate TD issues because it is much easier to connect with people¨
Team distribution
Team Communication
¨Working with cross functional teams provides an opportunity to expand knowledge in TD¨
Inter-team communication
¨Planning meeting is arranged in such a way that 80% of work will be related to product and 20% will always be dedicated to TD¨
Planning
Team Organization
¨TD reduces when developers choose a task themselves rather than it being assigned to them¨
Task Allocation/Distribution

Figure 2: It represents the way that data was coded. The interview answers are mapped into two themes: Team communication and Team Organization. The codes, team distribution and inter-team communication, come under team communication and the other two codes, Planning and Task Allocation/Distribution, come under team organization.

4 Description of the studied Organization

In this section, we will describe the categories and details of the companies we interviewed. The companies of the interviewees belong to various categories, ranging from startup to large multinational companies. We have categorized the companies based on the factor ‘staff headcount,’ which refers to the total number of employees belonging to that organization. Companies are categorized as Small size, Medium size, Large/Big size and Start-up.

- Small size[2] companies refer to privately owned partnerships or sole proprietorships whose annual revenue is less. The standard number of employees in this type of company could range from 10 to a maximum of 49 employees.
- Medium size[3] refers to companies that are larger than small businesses but smaller than big businesses and the number of employees ranges from 50 to 249 employees.
- Big/Large size[4] companies refer to big organizations that perform activities involving huge transactions and produce outcomes or products at a high volume. The number of employees is greater than 250.
- Start-up[5] refers to a special category of new companies that have started their initial business phase. They can be a small business, partnership or an organization without a fully developed business model and may not possess sufficient capital to proceed further with the development process with the only aim to achieve rapid growth in the market. The number of employees in a start-up is less than 10 employees.

4.1 Description of each Organization:

1. Company A is in the educational sector and is categorized under a medium-size company where we interviewed a developer working on web applications. The team consists of developers, QA, testers, and managers.
2. Company B is in the banking sector and belongs to a large size company. We interviewed four people from four different teams. Roles like tech lead, senior

[2] Small size company: https://ec.europa.eu/growth/smes/sme-definition_en
[3] Medium size company: https://ec.europa.eu/growth/smes/sme-definition_en
[4] Large/big size company: https://data.oecd.org/entrepreneur/enterprises-by-business-size.htm
[5] Startup company: https://www.investopedia.com/terms/s/startup.asp#:~:text=Startups

developer, developer, and developer who is also a solution architect. Tech lead has a team consisting of product owners, project managers, developers, and infrastructure people. This team works in a section called the integration of flow, which creates microservices to legacy systems in banks. The senior developer team has eight members in total. This team has both consultants and clients working on developing APIs in the cloud. Also, product owner, project manager, tech lead, scrum master and developers. The developer team has a team leader, developers, and a scrum master, who work on JAVA, Python and AWS. The developer and solution architect team are working with AWS especially in a mobile banking application. The team consists of a developer, technical architect, and tech lead.

3. Company C is in the publishing sector where they engage the customers by giving recommended articles based on customer preferences and belongs to a small size company. We interviewed a senior software engineer from the data platform team who works on ingesting data from various sources to downstream users. The team consists of 20 people. There are three sub-teams internally. They have a total of six members in a sub-team like a project manager, principal engineer, senior engineer, tech leads and junior engineer.
4. Company D is in the banking sector where we interviewed tech lead and developer from different teams and can be categorized under the large company. Participant of the tech lead team consists of five developers, one business consultant and one product manager where scrum methodology is used. The tech lead works on JAVA, and web technologies. Participants of the developer team consist of 5 developers, team lead also acting as a scrum master, business developer and product manager. The developer is working with the creation and enhancement of REST APIs.
5. Company E is in the IT financial service sector where it provides financial advisors to customers and clients and the company. We interviewed the tech lead. The team consists of developers, testers, product owners, a scrum master, and business analysts. Company E belongs to a large size company.
6. Company F is in the banking and finance sector mainly focused on investments related to items and assets. We interviewed a scrum master. They have a total of

seven members in a team, like developers, testers and UX designers. It comes under a large size company.

7. Company G is in the procurement software sector where we interviewed the product manager. It is a startup company. The team consists of two front end developers, two back-end developers, three QA and one product manager. The company offers solutions to users and makes the procure-to-pay process easy and efficient. The product manager we interviewed is involved in requirement gathering and analysis.
8. Company H is in the payment sector where mobile payment applications are designed for smartphones and can be categorized under the medium-size company. We interviewed a senior developer who works on deployment in the cloud, handling merchant related enrollment and life cycle. The team consists of developers, one technical manager, a tech lead and a product manager who will be working on multiple teams.
9. The company I is in an academic's sector where banking, financial advisory and legal services are provided for professionals and belongs to a big organization. We interviewed the project manager, working on development and project management. There are a total of seven members in a team like a project manager, developer, tech lead and a scrum master.
10. Company J is an IT consulting company where web-based applications are developed and uses AWS services and performs security support. The total team members are five which includes one technical lead and four developers. The size of a company is categorized under the large organization.

Table 1 gives a description of the interviewees involved in the study. Interviewee's role, total experience in the field, experience at the current company, size of the team in which they are currently working, development practices used by the team and origin of the company are described in the table.

Company	Role	Total-experience	Experience at the company	Team Size	Development practices	Country
A	Developer	8 years	3 years	10	Scrum	Norway
B	Tech lead	15 years	3 years	8	Scrum and Kanban	Norway
	Senior Developer	18 years	3 years	8		
	Developer	10 years	3 years	8		
	Developer and solution architect	14 years	4 years	8		
C	Senior data engineer	7 years	7 years	6	Scrum	Bulgaria
D	Tech lead	16 years	10 years	7	Scrum	Norway
	Developer	8 years	1 year	8	Kanban	
E	Tech lead	10 years	10 years	11	Scrum	US
F	Scrum master and business analyst	9 years	3 years	8	Scrum	India
G	Product manager	8 years	8 years	8	Scrum	US
H	Senior developer	14 years	1 year	11	Scrum	Norway
I	Project manager	6 years	6 years	7	Scrum and Kanban	Norway
J	Developer	2 years	2 years	5	Scrum	India

Table 1: It gives a description of the interviewees involved in the study. Interviewee's role, total experience in the field, experience at the current company, size of the team in which they are currently working, development practices used by the team and origin of the company are described in the table.

5 Results

This chapter will present what was found through analysis of the data collected. The results will be based on the answers to the research questions. First, types of communication used to discuss TD are analyzed, which describes the answer to the first research question. Next, the way team communication affects the TD occurrences are described using communication factors, communication of developers with customers and involvement of developers in decision making. These topics help to answer the second research question. And next is the research question based on team organization. Here we will describe the main factors of team organization, which causes TD mostly, the need for separate roles and dedicated teams particularly to TD, the impact of TD on the usage of common modules and task distribution to developers. We also focus on the significance of involving TD activities during planning sessions.

5.1 Team Communication and TD

5.1.1 Types of communication

We have classified communication into two types: Personal communication and group communication.

Personal communication

Personal communication is the communication between two individuals in a team or with the team lead. It is further divided into vertical and horizontal. Vertical communication is communication with the team lead or managers and horizontal is communication with other team members. Personal communication has both pros and cons. Pros include that it obtains a quick response to TD issues thereby saving time. On the negative side, personal communication details will be lost if a person leaves the company, and a particular developer might think that he/she is only working on TD if we have personal communication. An interviewee from company H said that *"A particular developer will feel like he/she is only working on TD issues if the communication is personal."*

Group communication

Group communication is the communication among group of individuals involving more than two persons and is divided into scheduled and unscheduled meetings.

We found that group communication is used more than personal communication to communicate when TD occurs, and scheduled meetings have more impact on reducing TD when compared to the unscheduled meeting. One of the main reasons to choose group communication is because everyone in the team should be aware of what is going on in a team or a company. The other reasons for choosing group communication include getting more insights and comments about TD issues as each developer will get a chance to share their opinions. Scrum activities like standup, planning, retrospective, review, and refinement are conducted as a part of scheduled group communication.

1. **Daily standup:** Almost all companies conducted daily standup meetings. Usually, the meetings took place in the morning and lasted from 10 to 20 minutes. Discussions regarding what has been done yesterday and what are the things that need to be done today are carried out. One of the interviewees from company D said *"Daily standup will be conducted for about 15 to 20 minutes. Discussions regarding what you did yesterday and what are you going to do today including TD issues."* TD issues faced by developers would be stored in JIRA and they will be discussed in daily standup meetings the next day. Issues will be selected as a task for a particular day based on the prioritization. An interviewee from company B said *"TD issues to be solved will be stored as a JIRA ticket and then quick review on these issues will take place in a daily standup. Based on the prioritization, TD issues will be solved."*
2. **Sprint planning:** Most of the companies had two weeks sprints and sprint planning was conducted at the beginning of each sprint. Some companies allotted specific time for TD communication in the planning session. An interviewee from company H said that *"We will discuss the next 2 weeks tasks in a planning session. Manager will plan in such a way like 80% of the work that we do in that planning session will be related to the product and the ongoing activities and 20% will be always dedicated to TD."* TD identified are saved as a ticket in JIRA by developers. In sprint planning, the tickets are reviewed and selected to solve based on the priority. Tasks and roles for

both ongoing and TD related activities for a particular sprint will be selected in this meeting. An interviewee from company E said, "*We will review all the tickets in JIRA during planning meetings and assign TD issues to developers based on the priority.*"

3. **Refinement:** Also called a grooming meeting. In this meeting, the team will make sure not to avoid TD by checking stories in the product backlog. Also, by making sure that all the information is updated in the product backlog to avoid not considering TD. The meeting will take place before sprint planning. TD stories would be picked in this meeting and there will be a discussion over the TD stories as well. An interviewee from company F stated, *"Refinement meeting will take place before the planning meeting. In this meeting, we will be picking up TD stories and we will discuss about the stories."*

In the retrospective meeting, details such as what goes well, what went wrong and what needs to be improved are discussed. In the review meeting, the demo and status of the completed work are shown. The above two meetings are not used by any of the teams for TD communication.

Table 3 represents different types of scheduled meetings conducted in each team to communicate. Brown cells indicate the meetings used by each team to communicate all kinds of activities and issues that arise in a team. On the other hand, green cells indicate the meeting used by each company to communicate all kinds of activities and TD related activities and issues.

Company	Interviewees	Scrum Meetings					
		Daily stand up	Sprint planning	Sprint review	Sprint retrospective	Refinement	Pre-Planning
A		🟩	🟩	🟧	🟧		
B	1	🟩	🟩				
	2	🟧	🟩	🟧	🟧		
	3	🟩	🟧		🟧		
	4	🟧	🟩		🟧		
C		🟩	🟩				
D	1	🟧	🟩			🟧	
	2	🟧	🟩	🟧	🟧		
E		🟩	🟧	🟧		🟩	
F		🟧	🟩	🟧	🟧	🟩	🟧
G			🟩	🟧	🟧	🟩	
H		🟩	🟩		🟧		
I		🟧	🟩	🟧	🟧		
J		🟩		🟧			

Table 2: It represents different types of scheduled meetings conducted in each team to communicate. Brown cells indicate the meetings used by each team to communicate all kinds of activities and issues that arise in a team. On the other hand, green cells indicate the meeting used by each company to communicate all kinds of activities and TD related activities and issues.

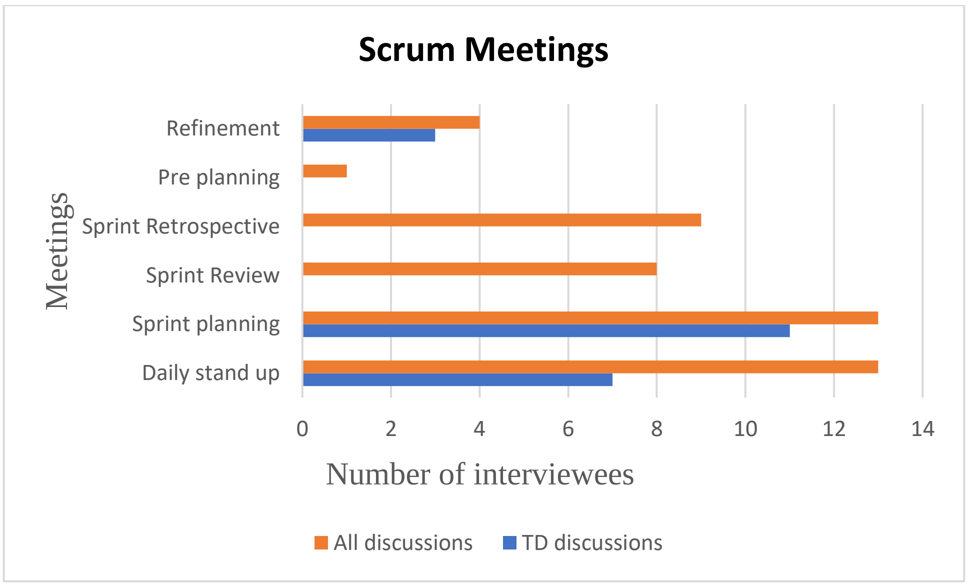


Figure 3: It represents the Scrum activities used by teams for general and TD related communication purposes. The brown color indicates the types of meetings used by interviewees for general communication purpose and the blue color indicates the types of meetings used by interviewees for TD related communication.

Figure 3 represents the Scrum activities used by teams for general and TD related communication purposes. Out of 14 interviewees, 13 said they have used sprint planning and daily standup for general communication purposes. Among 13 interviewees, 11 said they used sprint planning to communicate TD and 7 said they used daily standup to communicate TD. Next to these, 9 interviewees used sprint retrospective and 8 interviewees used sprint review for communication in general. But none of them used to communicate about TD. The refinement meetings are used by four teams for general communication in which three teams used this meeting to communicate TD. Pre-planning is used by only one team to communicate general communication purposes.

From Table 2 and Figure 3, we can say the discussions regarding TD and its issues are communicated mostly in planning sessions and daily standup meetings. In some companies, TD is also discussed in the refinement meeting.

Common Communication Channel

Common communication channels such as JIRA, slack, kanban board and sonarQube are used for TD related communication. JIRA is the most used tool to discuss TD issues. An interviewee from company E said that *"We use Jira ticket management tool where we break down all the features like requirements into epics, each has its own set of stories, and we will be working on those stories to complete the epic feature and then release the feature epic into production."* Next to JIRA, slack is being used for both groups and one to one conversation. SonarQube analysis is also used as a communication channel in some of the companies. SonarQube has a project dashboard where the results of the project will be displayed. Developers can access and modify the dashboard. Developers can easily understand where the major problems are on this project through the TD pyramid. The TD pyramid is a simple representation of the TD added to the system. An interviewee from company A said *"SonarQube analysis will tell us if there are TD in new pieces of code in which we are working. If any new TD occurs, flags will give a red signal saying that there is a new TD."*

Companies that use both scrum and kanban methodology use a kanban board as one of the communication channels to discuss TD related issues. An interviewee from company H, who uses both kanban and scrum, said that *"We use kanban board where we can see all those ongoing tasks, Td backlogs and work completed for a particular week."*

Modes of communication	**Types of communication**			
	Personal		**Group**	**Common communication channel**
	Vertical	**Horizontal**	**Scheduled**	
Daily Standup			X	
Planning			X	
JIRA				X
Slack				X
Developer to team lead	X			
Developer to developer		X		
Kanban board				X
SonarQube				X
Refinement or grooming			X	

Table 3: It describes the different types of communication used to discuss TD. Brown cells indicate the modes of communication used to discuss TD in different types of communication.

Table 3 gives the description of different types of communication used to discuss TD. Brown cells indicate the modes of communication used to discuss TD in different types of communication. Daily standup, sprint planning and refinement meetings under group scheduled meetings are conducted to discuss TD. Also, teams use personal vertical and horizontal communication to discuss TD, whereas group unscheduled communication is not used by the teams. The most common communication channels used by companies to discuss TD are JIRA, slack, sonarQube and kanban board.

Ways used to communicate factors of TD

Some of the interviewees also mentioned how specific TD factors are communicated in the organization.

1. **Risk of bugs:** This type of risk is communicated to both the developers and the product manager. As a developer, they know the importance and risk of bugs. But product managers might not know the importance of TD issues as they have the priority of adding new requirements and delivering them to customers. So, there is a need to communicate with product managers regarding the importance of TD, why it should be considered now and if not, what problems might occur. One of the interviewees from company D said *"I need to explain to the product manager why this is a risk of bugs. Also, I need to explain why it is important that we get to*

work with TD and its importance, instead of adding the new feature, or why this new feature needs to wait because we need to fix this first."

2. **Time constraints:** An Interviewee from company F said *"We do not have time to upgrade our back end or build the code or fix the TD. So, it is up to us how we convince the product team because there will be resources utilized as well."* If there are time constraints to deliver a project or need time to fix TD, the product team will be approached because it needs time as well as resources. 20% of the time in each sprint planning meeting will be allotted to addressing TD related issues.
3. **Cost:** There are primarily two methods on how refactoring costs are included in the team discussion. The first method is communicating with the project head and taking decisions and another method is communicating within team members with the help of the project head.
   An interviewee from company A said *"The one which is impacting our work will be prioritized and it will be communicated to the managers. Managers will discuss and decide whether to push this feature or not for this release."* The issues related to refactoring will be first communicated to the team head and a decision will be taken whether to push the task for the next release. If it is decided, managers will be contacted to do the refactoring. The refactoring decision will be decided by managers.
   Another Interviewee from company C said, *"Costs like Business cost, personal cost and refactoring cost are discussed and solved by the team with the help of principal engineers and product managers."* Here, cost-related issues are discussed by the team itself first. They roughly calculate the cost by asking several questions themselves like how much effort is needed for this task to be resolved, what are the impacts if it is untouched, and it is prioritized. All these discussions will be done with the help of product managers and principal engineers.
   An interviewee from company F said *"The architect and the senior developer will have a conversation regarding the risk, cost and time constraints of TD on the particular application. So, we'll have a high-level discussion over there."* Here in some companies, if any TD issues related to risk, cost and time constraints are

found, they will be saved as a product backlog in JIRA. Then it will be discussed with an architect and senior developers in a team. After the high-level discussions, it will be taken to the product team. The product team is the one who manages the product backlog and prioritization will be done. The final step is to inform other developers about the decisions taken for the TD.

5.1.2 Modes of communication used by co-located and distributed teams

Choosing modes of communication is an important aspect of an agile team for effective communication in teams. Table 4 represents the different communication modes used by each team to communicate TD related activities and other general activities in a co-located and distributed team. The blue-colored texts under co-located and distributed teams are used to communicate TD activities.

Company	**Participant**	**Modes of communication**	
		Co-located teams	**Distributed teams**
A	1	QA team, testers, **SonarQube**	**Google meet, zoom**
B	1	**JIRA, Kanban board, one to one**	**Video calls, Slack**
	2	**SonarQube, JIRA, Slack, one to one**	**Microsoft tools,** direct calls**, Slack**
	3	**JIRA, TD as a task in the sprint**	**Zoom, skype**
	4	**JIRA, one to one, Slack**	**Video calls, Slack**
C	1	**Slack, one to one**, whiteboard	**Video** or audio conferencing
D	1	**Slack, one to one**, whiteboard	**Microsoft tools, Slack**
	2	**Kanban board, Slack, one to one, JIRA**	Audio or **video calls**
E	1	**JIRA, one to one**	**WebEx, Skype**
F	1	**JIRA, one to one**	**Microsoft tools, zoom**
G	1	**JIRA, one to one,** ad-hoc meetings	**Skype,** Email**, Slack**
H	1	**JIRA, Kanban board, one to one**	**Microsoft tools, WebEx**
I	1	**Kanban board**	**Skype, zoom**
J	1	**SonarQube, JIRA, one to one**	**Microsoft tools, skype,** Email

Table 4: It represents the different communication modes used by each team to communicate TD related activities and other general activities in a co-located and distributed team. The blue-colored text under co-located teams and distributed teams is used to communicate TD activities.

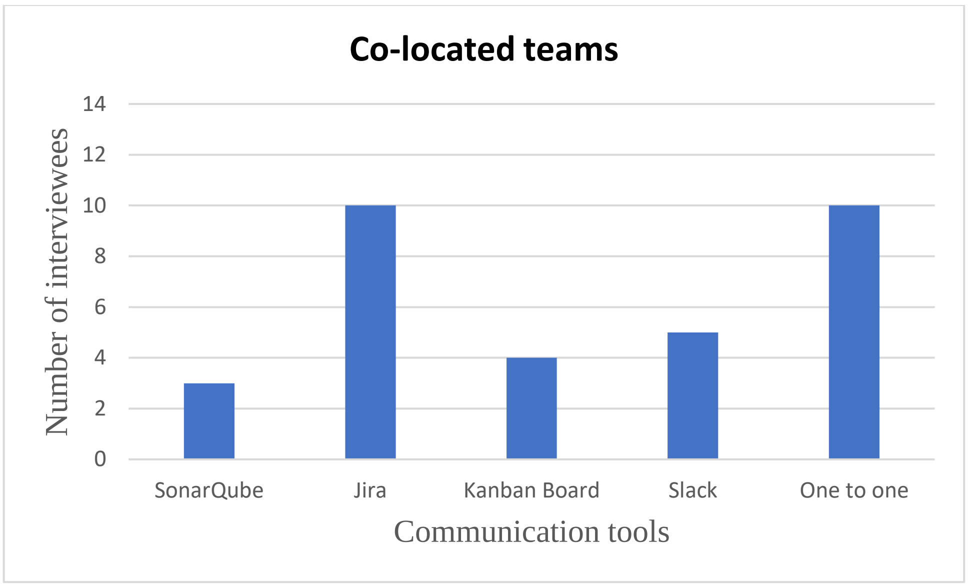


Figure 4: Blue color represents the different communication tools used by co-located teams to discuss TD. JIRA and one to one tool are used by 10 interviewees to discuss TD. Slack and kanban board are used by 5 and 4 interviewees, respectively. SonarQube is used by three interviewees.

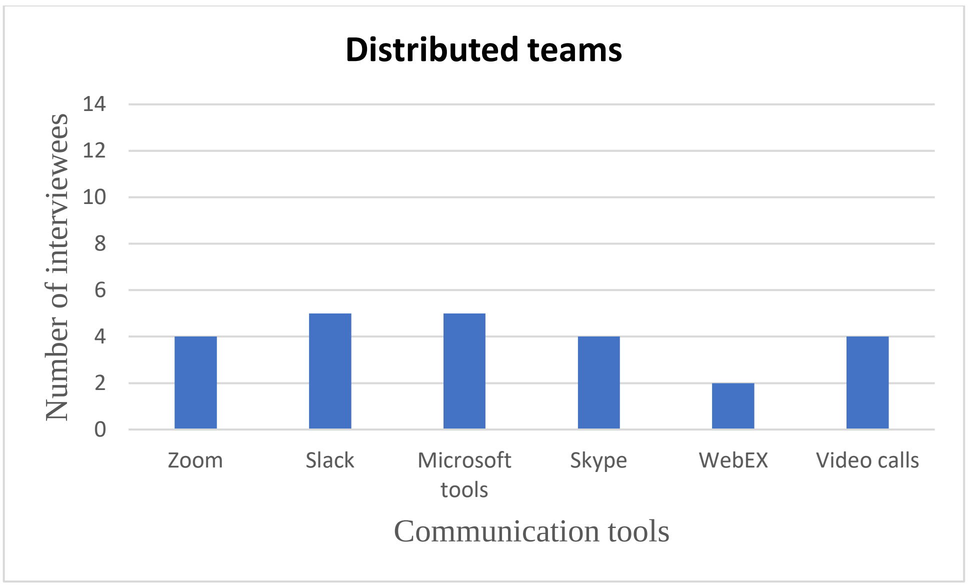


Figure 5: Blue color represents the different communication tools used by distributed teams to discuss TD. 5 interviewees used slack and Microsoft tools to discuss TD. Zoom, Skype and video calls are used by 4 interviewees and WebEx by 2 interviewees.

Figure 4 and Figure 5 give the description of communication tools used in co-located and distributed teams. It shows that most of the participants prefer one to one communication; JIRA and slack to communicate TD related activities when the teams are co-located whereas

Microsoft tools, skype, slack, zoom and video calls are mostly used to discuss TD activities when the teams are distributed.

Communication tools selected to discuss TD are based on the severity of the TD issues. If any TD needs to be solved immediately, then developers use one to one conversation with other team members or team lead. On the other hand, if the TD issue is of less severity, it will be added as a ticket in JIRA, or the team will use kanban board or slack to communicate.

JIRA:

JIRA is a ticket management tool where TD issues are stored as a product backlog, and it is used to track issues and bugs.

Kanban board:

Kanban board keeps track of the process flow while maintaining the number of work-in-progress activities. TD issues are also mentioned separately on board, which helps to keep track of all TD activities.

One to one:

If the TD needs to be solved immediately, developers have one to one conversation with other developers or team lead. It helps to solve TD issues fast compared to other communication tools.

Teams use slack, skype, Microsoft tools, zoom, video conferencing and WebEx to discuss TD related activities when they are distributed. Interviewees prefer to use video calls related tools because it is easier to discuss things face-to-face and by talking rather than typing. It also reduces the chances of miscommunication, which might increase the occurrence of TD. Another advantage of video conferencing is receiving a quick response from another person, decisions, and feedback do not get delayed which helps to solve TD quickly.

Slack:

Slack is used for communications with different parts of the company through channels. It is used to inform others about issues related to TD and other work-related tasks. Slack is used both by co-located and distributed teams to discuss TD related activities. Developers send messages to each other regarding TD issues and doubts on slack to get a quick response.

5.1.3 Inter-team Communication

Inter-team communication is communicating between the teams in terms of both co-located and distributed teams. So, findings in working with Inter-teams will also apply to some extent to team distribution. We found that working with inter-teams have both positive and negative effects on TD.

Most of the interviewees said they will learn new things about TD from other team members. Other members help to expand the knowledge database of TD. An interviewee from company J said, *“Other team people will bring specialized topic knowledge of TD which I might not be aware of.”* Other advantage that interviewees mentioned are if a team gets feedback of work from the other team, the TD issues can be found early and can be solved with the help of other team members.

One of the disadvantages of working with inter-teams is conflict in scheduling. The two teams should be free at the same time to schedule a meeting. This leads to a delay in a discussion of TD issues with the dependent teams. Delay in the TD discussion would accumulate more TD in a project, which in turn affects the quality of the project. Next is conflict in decision making about TD issues. Each team will have a different priority which leads to bias in prioritizing and decisions about TD. This results in conflicts between the team and the delay in solving the issues. Other reasons are that they should be dependent on another team. For example, even though one team has finished their work, they must be dependent on the other team to complete their part. This will lead to a delay in finding out the TD issues and solving them. An interviewee from company B said, *“It can be a challenge if the teams don’t know about what other teams are working.”* Both the teams must have some knowledge on TD of what they are facing to make the communication more efficient. Figure 6 shows two main issues in inter-team communication while working with TD.

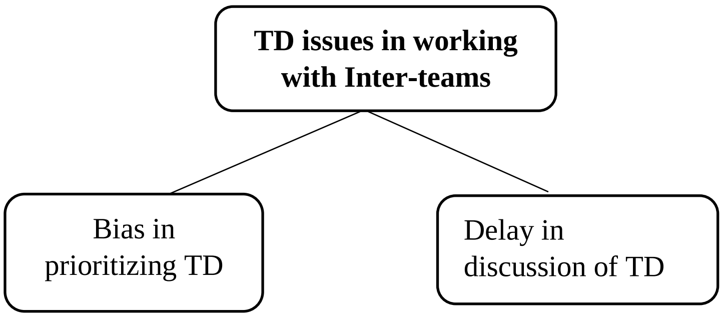

TD issues in working with Inter-teams
Bias in prioritizing TD
Delay in discussion of TD

Figure 6: Two main issues in inter-team communication while working with TD are biased in prioritizing and delay in the discussion of TD issues.

5.1.4 Intra-team communication

Intra-team communication is the communication done within the team and it can involve both co-located and distributed teams. Our questionnaire was structured in a way that interviewees answered about the positives and negatives of inter-team communication by comparing it to intra-team communication. So, it can be presumed that intra-teams have positive and negative effects that are opposite to inter-team communication.

Some of the advantages of working with an Intra-team are conflicts in taking a decision and scheduling a meeting will be less when compared to inter-team communication. For example, arranging an unscheduled emergency meeting within a team without depending on the other team is easy since all members of a team are generally available at some common time. In inter-team communication, each team will have a different priority in deciding and managing TD when compared to intra-teams where there would be fewer different interests. So, deciding within a team leads to less conflict in decision making.

Some of the disadvantages are that knowledge sharing will be limited compared to inter-team communication involving more than one team. If members in a team discuss TD activities and issues with other team members, they might get extra knowledge about the concepts which they might not know. Also, it is possible to solve the TD issues quickly if two teams work together by sharing ideas.

5.1.5 Team distribution

Team distribution refers to the extent to which team members of software development projects are separated from each other (distributed) or close to each other (co-located) in terms of the physical location.

We found that co-located teams are preferred by team members to discuss when TD occurs. Interviewees found it was easier to communicate TD and connect with the team members when they are sitting nearby. An interviewee from company F said, *"It is much easier to connect with people in a co-located team. It is efficient and important that you have a person nearby to communicate issues related to TD*." Second, the discussions will be shorter and crisp, and people will feel comfortable having TD related conversation face to face in the workplace. Team members in co-located can communicate efficiently due to physical presence by using co-located office spaces such as white-boards and other informal communication such as unscheduled meetings which helps to make a quick discussion whenever TD occurs.

Company F and G are working with companies in different countries. So, they have a difficulty in time zones, and language barriers in communication, which might cause miscommunication in expressing TD issues. For example, company G is working in the US. They have people in other countries which have different time zones. There will be at least 9 hours' time difference, which will be difficult to schedule a meeting where all team members can participate. Also, the team members who are not native English speakers often have difficulty while communicating in English with native English speakers which causes the meeting to get delayed and views about TD issues cannot be expressed efficiently.

The amount of verbal communication is less in distributed teams, and this leads to misunderstandings and confusion among team members, which lead to an increase in TD. Videoconferencing should be preferred when direct face-to-face communication is not possible because it most resembles natural communication. Figure 7 shows two main issues in a distributed team while working with TD.

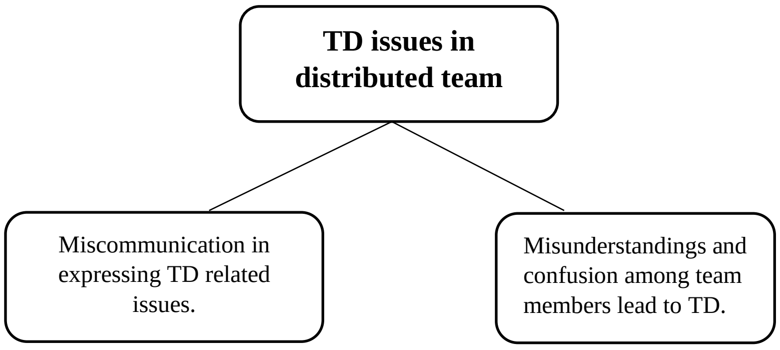

TD issues in distributed team
Miscommunication in expressing TD related issues.
Misunderstandings and confusion among team members lead to TD.

Figure 7: Two main issues in a distributed team while working with TD are that miscommunication of TD issues and misunderstandings among team members lead to the occurrence of TD.

5.1.6 Team size

Team size is defined as the number of team members on the development team. Since large teams are more likely to be distributed and smaller teams are more likely to be co-located, findings in the team size category would be correlated somewhat to findings in team distribution.

Our results indicate that in general, a small team of developers helps to reduce TD. For instance, an interviewee from company A said *"If you are following an agile principle, smaller teams are better in terms of communication and coordinating TD. It is easier and more effective."* It is easy to communicate and manage TD if there are a small number of developers in a team. Also, it will be easier to know each developer in a team, the tasks they are working on and their performance. The number of necessary communication links will be lesser if the team size is small, which enables more effective communication of TD.

Further, our analysis reveals that the addition of team members leads to both positive and negative effects on TD. Teams must choose their communication mechanisms wisely in larger development situations as otherwise, communication may be delayed. Since any addition of a team member requires more coordination to communicate, it can cause more TD. This might make the developer to have less knowledge about the existing TD due to lack of documentation. Also, if a new developer with less experience joins the company, he might unintentionally introduce a new TD. New employees can accumulate TD due to less understanding of the working pattern in a team. An interviewee from company G said, *"One* *of the issues that caused TD is new people joining the team with less knowledge about the code which results in making them use shortcuts."* On the other hand, there are also positive effects to it, particularly, if people in a team are knowledgeable, well-trained, and well-equipped. In that case, a larger team will help to solve tasks quickly and each developer can pick one task and solve it. In this way, a developer will have more knowledge about the chosen task which helps to reduce TD. An interviewee from company J summarized by saying *"If there are more people in a team, each one can pick a task and if an issue occurs it can be easily identified because each one knows their task well."* Figure 8 shows two main issues in a large team while working with TD.

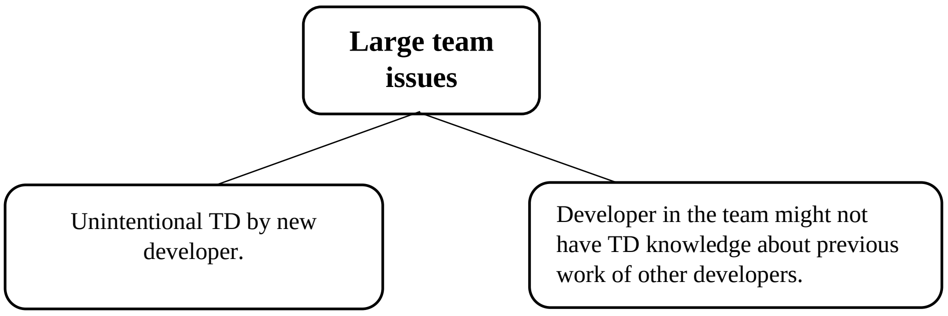

Large team issues
Unintentional TD by new developer.
Developer in the team might not have TD knowledge about previous work of other developers.

Figure 8: Two main issues in a large team while working with TD are that new developers create unintentional TD and have lack of TD knowledge about previous work of co-workers.

5.1.7 Customer collaboration

We found that close collaboration of developers directly with customers helps to reduce TD. Requirements are considered as the main element in developing a product and it will be constantly evolving due to changes by customers. In most of the companies, interviewees said requirements from customers are collected and updated by the team lead or project manager and it is stored in JIRA. There is a possibility to miss the details while communicating them to developers. Missing details of requirements can lead to a type of debt called requirement debt. Requirement debt is referred to as compromises made concerning what requirements the development team need to implement or how to implement them (Alves et al., 2014). The probability that a developer will miss requirement details is reduced when they have a direct conversation with customers, thereby reducing TD. One of the interviewees from company B said, *"We will get to know things we want to do in future earlier which helps us to improve our planning which helps to reduce the occurrence of TD."*

The other reasons include early customer feedback and future details of what customer needs helps to improve sprint planning while having direct feedback enhances the management of TD. Also, it will help to improve the product quality and reduce TD by explaining the issues to customers and obtaining more time to deliver the product.

On the other hand, some of the developers said it may not have an impact on TD. Knowledge about TD related issues is generally lacking in customers and hence it might not be useful to communicate with customers regarding TD related issues. One interviewee from a company F said, *"Developers and customers communication is not necessary. Technical people know more about TD compared to customers."* Some companies think that TD mostly arises from developed or finished products which we have already delivered to customers. It might help receive precise requirements while working on a project but not in reducing TD, which already occurred.

5.1.8 Developers participation in TD decision making

Developers are mostly involved in giving suggestions to take a TD related decision rather than being directly involved in taking a decision. The involvement of developers in obtaining suggestions helps to reduce TD and efficiently manage TD.

- In Company A, the team member suggestions would be considered if he/she works on a particular code which has TD.
- In Company B, if TD occurs, it will be an open discussion with the architect and input from developers is taken into consideration while making decisions.
- In Company C, suggestions will be taken from developers, but the deciding authority will be the principal engineer or a product person.
- In Company D, developers are involved in decision making based on the severity of the issue. If it is related to cost, the decision will be taken by managers and team lead without considering developers' suggestions. Prioritization will be done by developers' suggestions.
- An interviewee from Company E said that TD prioritization and other decisions will be done in a team discussion where the whole team will review each subtask and solutions from team members are considered.

- In Company F, if TD occurs, the discussion will be done with developers and their suggestions and statements will also be considered and then the architect will decide. An interviewee from the company said, "*Every team member has the right to give a statement and they have to agree to it."*
- In Company G, they have a norm that each developer will have to come up with a list of highest priority TD items with reasons. Based on the list, decisions will be taken by the manager level.
- In Company H, a brainstorming session with developers is conducted to identify TD. Then the prioritization of TD will be done by obtaining suggestions from them. During prioritization, the managers might have discussions with other teams. However, the final decision will be taken by the manager.
- In Company I, flat hierarchy is being followed where the developers will have the liberty to come up with innovative solutions to TD issues and suggestions for prioritizing TD is also obtained from them.

On the other hand, an interviewee from company J who has two years of experience, said that junior developers are not involved in decision making. They will ask for suggestions based on the experience of the developer. Only the senior developers who have more experience in the field will be involved in giving suggestions to decide TD. Here, the junior developer will report work to senior developers so that the senior developers would be aware of the work done by them. So, it does not affect TD occurrence.

As we can see, most of the companies think involving developers will help to make decisions regarding TD. Also, it helps to reduce the TD as all the developers will know the severity of each TD issue and the importance of each TD will be analyzed to make a decision.

5.1.9 TD documentation

Documenting everything that needs fixing or updating and has been fixed is a crucial step for developers and other people in a team to know that there are issues that need to be fixed. It helps to minimize time and effort to locate and fix issues. We found that only a few companies use the documentation for TD. Those companies documented the entire task along with the TD issues without separate documentation for TD. The most important tools used for documentation in JIRA. JIRA works as a TD backlog and is used by developers to make TD more visible. One of the interviewees from company B said *"We will document the codes, sequence diagrams in JIRA. If you need inputs for other projects, we will refer to the documentation. We use the same for TD related issues."* Other tools used as documentation are SonarQube analysis, README files, wiki pages and MS files. SonarQube is mainly used to check code-level issues, core level performance issues or issues that arise in java code. One of the interviewees from company A said, *"SonarQube keeps track of TD and debt trend analysis will keep track of whether we are clearing TD or not."*

However, maintaining and updating documentation is one of the issues mentioned by the interviewees. Outdated documentation might lead to the occurrence of TD (Brown et al., 2010). If the changes are made in programs without accompanying changes to documentation, the resulting mismatch in documentation is called documentation debt. The TD list must be updated after each release when items should be added as well as removed (Guo & Seaman, 2011). The next issue is the lack of required information in the software documentation that may lead to documentation debt (Farid, 2012; Soares et al., 2015).

5.2 Team Organization and TD

Here we describe the main factors of team organization, which cause TD mostly, the need for separate roles and dedicated teams, particularly to TD, the impact of TD on the usage of common modules and task distribution to developers. We also focus on the significance of involving TD activities during planning sessions.

Organization	Factors causing TD				
	Inappropriate planning (deadline)	Lack of well-defined process	Lack of knowledge	Lack of Training	Employee turnover in the team
Company A	❌	✅	❌	❌	✅
Company B	✅	❌	❌	✅	❌
	✅	❌	✅	❌	✅
	❌	❌	✅	❌	✅
	✅	❌	❌	❌	✅
Company C	✅	❌	❌	❌	✅
Company D	✅	✅	✅	✅	✅
	✅	✅	❌	❌	✅
Company E	✅	❌	✅	❌	✅
Company F	✅	✅	❌	❌	✅
Company G	✅	✅	✅	✅	✅
Company H	✅	❌	✅	❌	✅
Company I	❌	✅	✅	✅	✅
Company J	✅	❌	✅	✅	✅
Total	11	6	8	5	13

Table 5: It describes the factors of Team organization that cause TD. Green tick and red cross represents a factor that is responsible and not responsible for the occurrence of TD respectively in a software company.

5.2.1 Main Factors affecting Technical Debt

1. Employee turnover in the team

From our analysis, employee turnover in the team can be considered as a major factor causing TD, which refers to the percentage or number of employees leaving a team from an organization and being replaced by new employees instead. Under this, several reasons for causing TD can be taken into consideration such as when an experienced person is leaving a company and a new person joins, there will be a lack of knowledge if the former person does not document his tasks properly; therefore, TD occurs since the knowledge sharing process needs to be done repeatedly which slows down the development process that affects the timeline of the delivery.

An interviewee from company H expressed *“The change in team members will affect the velocity of how it can solve TD on time; it will slow down the process; if new developers are coming, and they need to learn about old systems and new systems.”* So, when a person leaves an organization and a new person joins, it would probably take time to understand the working environment and the ability to deal with existing TDs in the project. The development plans and development costs need to be continuously changed due to the change in team members, thus lose clarity and there are chances of messing up with the development process, which causes TD. The Technical Lead from the company E expressed *“The change in employees in the team sometimes causes TD, when the new person joining the team may not have a good overview of the application.”*

2. Inappropriate Planning

Sprint planning should be done properly if the process is not planned properly, and development starts without seeing the entire solution, it makes it very difficult to proceed with the development process. The deadline also comes as a major factor where close deadlines lead to improper planning of the process, which finally leads to TD. In some cases, even if TD is identified, it may not be resolved on time, since planning has not been done properly, which increases the existing TD.

3.Lack of well-defined process

The customer may not always be accurate about the requirements they need, which leads to a change in the specifications of the product. An interviewee from company A expressed, “A *well-defined process helps code clean-up, no accidental merging, 7/8 checks quality checks, GitHub actions, SonarQube inhouse GitHub checks which limits TD from a new developer.”* When the development process is not defined properly, there are chances of TD occurrence, and this could be a collaborative failure during the development process when a team implements a feature without thinking much about how to build from an implementation point.

4. Lack of knowledge

The requirement gathering should be done in a proper way to deliver the product accurately. An interviewee from company H expressed TD arises *"where some team or team members may be a team which is outside our team who will try to do some shortcuts to deliver a product because they don't have much understanding of the existing system or existing design."* If one big team splits into sub-teams for different domains, the team members may not be familiar with and have an overview of everything they do. The pressure of the team to deliver the product adds more TDs when they work with their half-knowledge. Another interviewee from company J expressed *"If there is no senior member in the team then it results in lack of knowledge."*

5. Lack of training

In most organizations, there is no training provided particularly to TD. The team members agree on and follow standard code conventions on how to code. An interviewee from a company I expressed *"It is important that knowledge of sharing so that we can eliminate lack of training concept."* The sessions provided by senior team members to the newly joined developers overcome the lack of training.

5.2.2 Additional Factors causing TD

The following factors are obtained from the interviewees, which are considered as some other factors that cause TD. These factors may hardly occur in software teams and result in TD.

- **Change in technology**

  An interviewee from company D expressed *"Change in technology from one front end framework to another."* TD exists in a codebase that was written using legacy technologies.

- **Change of coding style**

  TD arises because of a new feature and when someone has a better opinion or suggestion, which is more effective for the development process.

- **Culture and Handling ambiguity**

  An interviewee from a company I expressed *“It is the comfortability of a person in handling uncertainty.”* TD can arise in teams when the team members might not understand the flow of application to its full extend.

- **Quick-fix solutions**

  The fast release of what has been done and a quick fix of solutions results in a lot of TD.

5.2.3 Managing the Team change

When an experienced or senior person leaves the company, it often results in TD. This can be managed by:

- **Better Code quality**

  The review of code must be done to ensure code quality and improve knowledge sharing. It also helps to eliminate some amount of TD that is caused due to a lack of knowledge. Peer reviews are done in organizations so that TD can be managed. SonarQube is a tool that is used by organizations to measure the quality of code in terms of standard and bug-free code, which may help to resolve TD that occurs due to non-standard codes. The Quality Assurance (QA) team monitors the existing TDs and the TD cleared so far. An interviewee from company B expressed “*We have a pretty good overview of all code we have written even if a person leaves then some other guy on the team has also worked on that system.”*

- **Managing Documentation**

  If proper documentation is maintained when a person leaves the organization, the TD can be monitored and tracked easily. The tasks performed by that person can be documented in wiki pages for further reference. An interviewee from company B expressed *“If a person leaves from a team, we document the work status in JIRA.”* This way of documentation helps to understand the ongoing or in-progress tasks for both the existing and new employees in a better way and this gives a good understanding of code written by the developers.

- **Conducting Knowledge Transfer (KT) Sessions**

The new person joining the organization requires adequate knowledge sessions, KT, which includes an introduction about the project, domain, and the applications they are going to work on. This makes them comfortable and familiar with their tasks. This helps to reduce TD. Sometimes, experienced developers give KT sessions and explain the tasks before assigning them to the new developer.

- **Organizing Training**

Some organizations provide training to newly joined members which they think reduces TD. An interviewee from company B expressed *"We'll be giving a test coverage increase task to a new person so that he will know the code base before he starts mugging the domain-related activities."* They have wiki pages that contain information about the project. This helps the new person to understand the project and domain, which is considered as training.

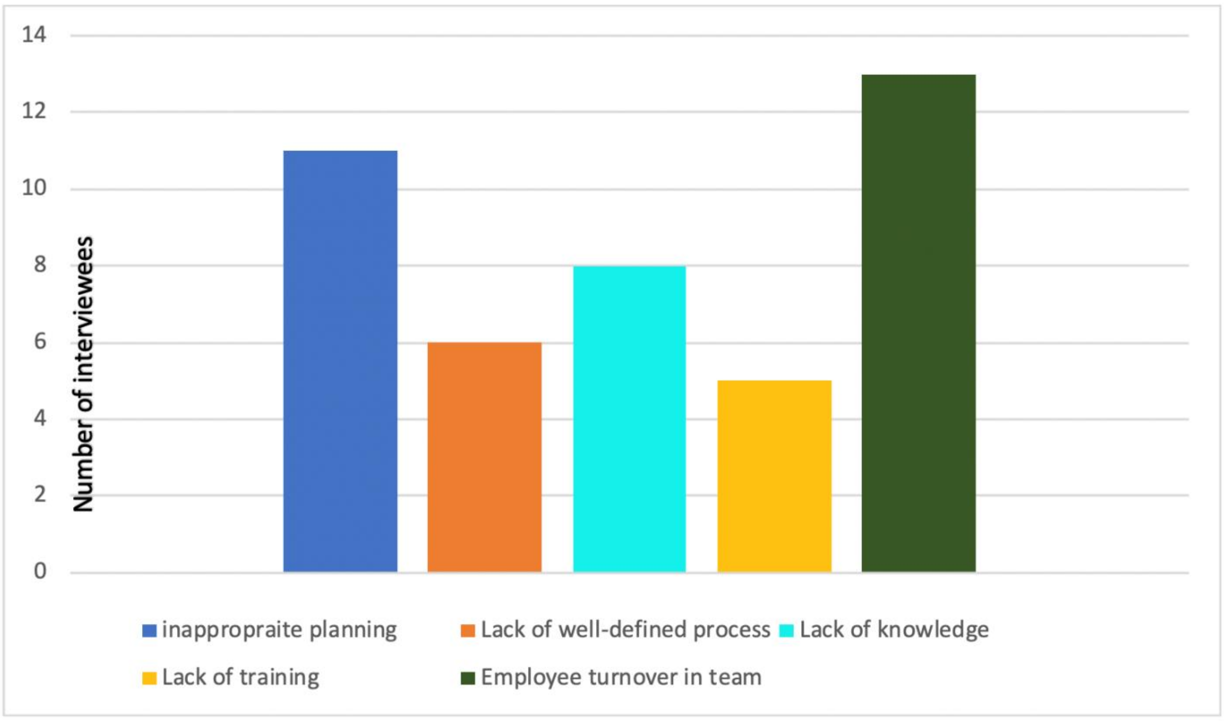


Figure 9: Each color represents a team organization factor that causes TD. Out of 14 interviewees, 13 answered employee turnover in the team as a major factor of TD, 11 answered inappropriate planning as a factor, 8 answered lack of knowledge, 6 answered lack of well-defined process, and 5 answered lack of training as cause of TD in their teams.

Out of the 14 interviewees we interviewed, from Table 5 and Figure 9, we found that employee turnover in a team is the primary cause for TD, then inappropriate planning, lack of knowledge, lack of well-defined process and finally, lack of training. Very few interviewees shared other factors of TD such as business-critical requirements, time constraints, change in code/technology, change of coding style, culture and handling ambiguity and quick-fix solutions.

5.2.4 Specific Roles and Separate team for TD

As a result of our analysis, TD-specific roles do not exist in most of the organizations that we interviewed. An interviewee from Company E expressed *"Handling TD is considered as the shared responsibility among product owners, product managers, business analysts and development teams."* A ticket system like JIRA is used to create, define, track and record tasks and is assigned to different roles. An interviewee from company F expressed *"There are roles of TD management, TD is found in two ways: Architect, who comes up with TD and via team also since they are closer to the code they are working."* Another from company G expressed, *"Ours is a small company and we are looking to grow, to get more market revenue, so TD is often put aside. We know that we have a lot of TDs because of quick-fix solutions, just release whatever we can quickly."*

Most of the organizations do not possess any separate team for managing TD because they suggest that the one who works closely with the application may know the functionalities of that application better so the entire team should work for TD. An interviewee from company G expressed *"We very recently formed a team called platform team that can work on TDs goes" but the interviewee was not able to discuss its impact on TD management since only the team has been formed but not fully functional.*

5.2.5 Task Distribution

The task distribution to the developers is usually performed during planning meetings and is done in two ways: assigned or unassigned. In unassigned task distribution, developers pick their tasks in which they are comfortable and interested; and in assigned task distribution, the tasks are assigned to the developers by their Scrum master, project manager or technical lead.

- **Tasks Picked by Developers**

  An interviewee from company E expressed *"Developers can pick their tasks, but they cannot pick low priority tasks related to TDs leaving the high priority ones, they can pick tasks from high priority as their wish or interest."* During sprint planning, backlogs of all tasks are maintained, and developers pick tasks that are more relevant and can be performed within a short time. An interviewee from company B *expressed "TD can be reduced if we pick up tasks ourselves and if there is a set of TD tasks which I can choose, I will choose which I am familiar with."* When developers pick their tasks to be performed, they choose tasks in their interested area, which helps to perform the development process faster and reduces TD since they have good knowledge in the area.

- **Tasks Assigned to Developers**

  In some organizations, tasks are being assigned to the developers. There are some factors taken into consideration while assigning tasks: proficiency of developers, project or domain experience, comfort level, complexity, the effort required and prioritization of tasks. An interviewee from company E stated when tasks are assigned "*It's good to have a third eye to observe the things because when one member in a team fails to notice and if they find TD, it can be improved, improve overall application stability and maintainability.*" When less experienced developers with less domain knowledge are assigned tasks with more complexity in that domain, it may create TD.

In general, from our analysis, in most organizations, developers pick their tasks themselves which helps them to reduce both the development time of the project and TD.

5.2.6 TD during the planning

About half of the organizations, we interviewed considered TD during their biweekly planning sessions and sprint planning. An interviewee from company E stated they *"We have release planning in every quarter for next quarter, where they review all TDs and assign based on priority and decide the TDs that can be solved in the next quarter."* TD is discussed in usual planning meetings where they will cover all epics which includes TD, any new feature request, and bug fix.

An interviewee from company H stated, *"In planning meetings (biweekly using JIRA), manager or engineering manager plan in such a way that 80% of the work that we do in that planning session will be related to the product and 20% will always be dedicated to TD."* They prefer JIRA to discuss TD stories. Since TD is discussed during the planning meeting, every team member can participate, and everyone is made aware of the status of TD. They can document it, and it can be used for future reference when similar TDs occur.

An interviewee from company B stated, *"We have a day reserved only for technical work on Thursdays called technical Thursdays where we normally do not perform some feature things, and all will work on technical tasks."* Very few organizations do not consider TD in their planning sessions as an interviewee from company A stated, *"We do not consider TD during planning, but whenever a piece of code is touched, we eliminate TD there, this is the strategy."* The planning meetings mostly include discussions with the development of projects.

From our results, we found that TDs are discussed mostly during planning meetings. In biweekly planning meetings using JIRA, managers plan to discuss 80% of work-related to product and 20% of their work will be dedicated to TD. The sprint planning also considers TD based on the priority of the team and TDs with high severities that need to be resolved immediately.TD is also discussed in support projects and is communicated among all team members that help all the members of the team aware of the TD and document it and ask everyone to follow those steps when TD occurs. The release planning is done every quarter for the next quarter where TDs are reviewed, assigned, and decided to resolve based on priority.

5.2.7 Usage of Common modules

All organizations have common modules or reusable components, which are common and can be used by multiple teams. Most organizations use libraries, codebase, and infrastructure in common. The list of components used in common, their advantage and how they impact TD is shown in Table 6.

Reusable / Common Modules	Description	Impact on TD
Libraries, Third-party libraries, Common repositories	No need for code duplication, Review the changes being done	Decreases TD
Common guidelines for logging, building blocks for applications	Consistent way across a team, a culture of sharing and improving standards	Decreases TD
The common infrastructure codebase	A well-processed way is developed, turn things more efficient way	Decreases TD
Reusable codes, standard development procedure, standard delivery procedure	Increases speed of development reduces the number of errors, save time in creating a new component of function every single time, brings sustainability,	Decreases TD
Internal common codes which are produced and published internally and used by other teams also	Quality code can be shared with others, do not need to re-invent and write from scratch	Decreases TD
Centralized piece of code	The modifications on these modules may affect other teams which makes them less controllable to manage those common modules	Causes TD
Common features in applications	It does not create any effect on TD by using common modules	May not cause TD

Table 6: The usage of common modules and their impact on the occurrence of TD. The first column indicates the most common reusable modules used in various organizations; the second column indicates their detailed description, and the third column indicates their impact on the occurrence of TD.

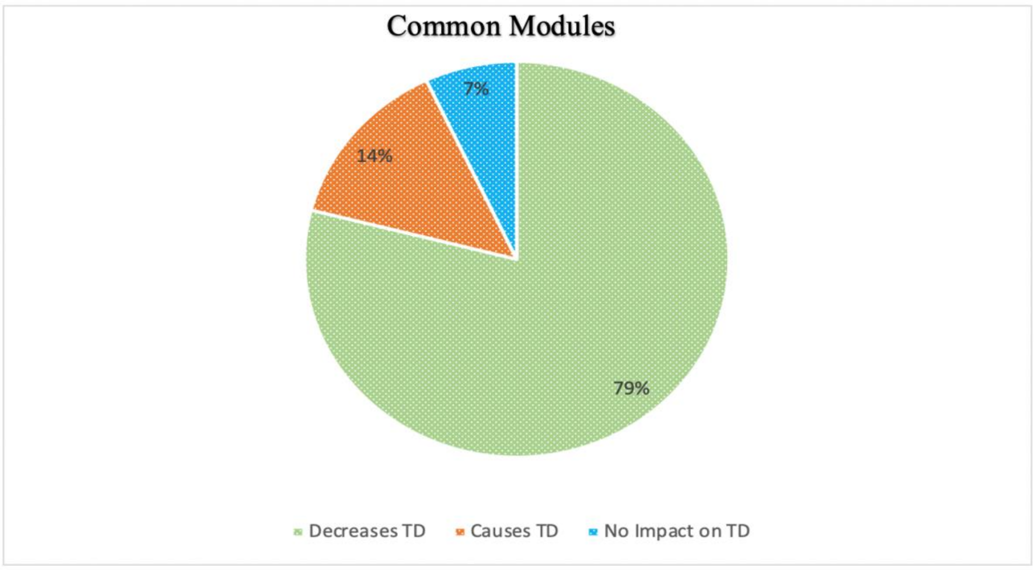


Figure 10: It represents the impact of usage of common modules on the occurrence of TD. Green color (79%) indicates usage of common modules decreases TD, orange color (14%) indicates usage of common modules causes TD and blue color (7%) indicates no impact on TD.

From Table 6, we found that the usage of common modules in the organizations can be categorized into three groups: Firstly, we found that the reusable components used in the project help the developers to reduce TD. Secondly, in some organizations, the usage of those components causes TD. Finally, the usage of common modules may or may not have an impact on TD. From our analysis, we identify that there is a decrease in TD when organizations use reusable components in the projects. From Figure 10, we also found that TD decreases in most organizations by the usage of common modules. Only a very few percentages of organizations have no impact on TD.

6 Discussions

This section will discuss the results presented in Chapter Results. First, the results will be used to discuss the answers to research questions, which is followed by a discussion on contribution, threats to validity, and the limitations of this study. Mitigation practices provided in this chapter are based on background studies and from our knowledge and experiences.

6.1 Team Communication

Effective team communication is a fundamental requirement for the teams using agile methodology. Picking the best communication option to discuss the occurrence of TD and its related issues is a crucial factor to share information. This section will discuss the following research questions:

R.Q 1: How do software teams communicate when TD occurs?

TD is communicated based on the seriousness of the issue. First, if TD affects the ongoing activity and needs to be solved immediately, then it is communicated in two different ways:

1. Communication will be one to one personally with the team head, which leads to a discussion in a spontaneous meeting.
2. Teams use slack to communicate the occurrence of TD so that everyone in a team knows that there is a TD that needs to be solved immediately.

Second, if occurred, TD does not affect any ongoing activities; it is communicated in two different ways:

1. Teams use product backlog in JIRA. TD will be stored in JIRA as stories or epics in the product backlog. Each day, stories will be picked randomly, and factors of selected TD like risk, the cost, time constraints and other effects will be discussed, and then the team will proceed to solve the TD.
2. Next, TD will be communicated in sprint planning meetings. The meetings will be held in a way like 80% of the work in the planning session will be related to the ongoing activities and 20% will always be dedicated to TD discussions. If there are more TDs to be solved, 20% will be increased to discuss the TD. If TD must be solved on a particular day, then it is discussed in daily stand-up meetings.

R.Q 1.1: What are the types of communication used to discuss TD and its factors?

From our findings, we found that group communication has a more positive impact on TD compared to personal communication. Each member of a team will know the progress of the work if it is group communication. Also, it is possible to get more details and comments from developers regarding TD and its issues.

From our findings in the Figure 3 results, we found that discussions regarding TD and its issues are communicated mostly in planning sessions and daily standup meetings. In some companies, a refinement meeting is also used to discuss TD activities.

Daily standup

The goal of the daily standup is to get insights into team members' tasks and solve any issues in the way of making progress. The norm in the daily stand-up meetings was that the team lead asked for status for each of the tasks in the sprint, which resulted in little coordination in the meeting (Nyrud & Stray, 2017). Tasks need to be solved are taken from JIRA and all the TD issues are discussed. Along with TD issues, the larger the team, the less is the satisfaction with Daily standup meetings. Large teams using this meeting should therefore pay special attention to improving the quality of these meetings (Stray et al., 2017).

Sprint planning

The goal of the Sprint planning meeting was to break issues down into smaller tasks and allocate them to the team. This meeting has other purposes like estimating time on each task, discussing requirements, and talking about whom to get in touch with if more information was needed. Sprint planning is considered an important activity to discuss TD. TD issues are discussed and selected based on priority by considering factors like cost, risk and time constraints and they are assigned to the developers. We found that some companies allotted separate time in a planning session to discuss TD. For example, if 80% of the time is spent discussing ongoing activities, the remaining 20% will be exclusively for TD discussion. We also found that if the team has more TD to be solved, 20% of the time might get increased.

Refinement meeting

It is also called product backlog grooming. To avoid not considering TD, stories or tasks in the product backlog are checked and it is made sure that all the information is updated in this meeting.

From our findings, we also found that a quick response to an issue can be obtained in personal communication. However, it also has some issues like details will be lost if someone leaves the company and a particular developer might think he/she only works on TD. If group activities are used to discuss rather than using personal communication, these kinds of issues can be avoided.

From Table 3, we can say, in a group scheduled meetings, daily standup, sprint planning and refinement meetings are used to discuss TD. If TD needs to be solved immediately, teams use personal vertical communication that is discussing with the head or team lead to get a quick response. Also, teams use personal horizontal communication, discussing with other co-workers to communicate TD. Group unscheduled communication is not used by the teams to communicate TD. The most common communication channels used by companies to discuss TD are JIRA, slack, sonarQube and kanban board.

Communication tools

From Figure 4, we can say that communication tools such as One to one, JIRA and slack are most widely used in co-located teams to communicate TD activities**.** One to one communication is preferred more compared to other tools because to get quick response considering JIRA and slack where the waiting time is more comparatively. Also, teams use the Kanban board to keep track of TD issues.

From Figure 5, we found Microsoft tools, skype, slack, zoom and video calls are preferred to communicate in distributed teams. Face to Face video calls is preferred by the team when they are distributed to avoid miscommunication.

We also found some of the specific TD factors and the way the factors are communicated. Table 7 explains how TD factors are communicated in a team. The risk of bugs is communicated to both developers and product managers. Cost related activities are communicated in two ways: first, they are communicated to the team head; secondly, the team members themselves will calculate the cost with the help of managers. Other ways to communicate risk, cost and time constraints using JIRA.

TD factors	Ways used to communicate
Risk of bugs	Communicated to both developers and product managers. But product managers need more explanation regarding the risk of adding new requirements which leads to extra bugs in an application.
Refactoring, business, or person cost	1. Communicated to team head whether to refactor or not
	2. The team members themselves calculate the cost, risk, effort needed for TD with the help of product managers
Risk, cost, and time constraints	TD is stored as a backlog in JIRA. The senior developer team will have a high-level discussion. The product team will prioritize and inform developers regarding the decision of which TD to solve.

Table 7: It describes the way that factors of TD such as the risk of bugs, refactoring cost, business cost, and person cost are communicated.

R.Q 2: How does team communication affect the occurrence of TD?

Team communication plays a vital role in causing TD occurrence. Teams should consider team composition, choosing the right modes of communication and the way the teams communicate with other members to reduce the occurrence of TD. In addition, communication with customers and developers involved in decision making can also have an impact on TD. This can be discussed further and summarized more in detail in RQ 2.1, 2.2 and 2.3.

R.Q 2.1: How do communication factors impact TD?

The communication factors considered Team size and Team distribution. Communication regarding intra-team and inter-team are also considered.

Inter-team communication

Agile teams are often embedded in a large organization and dealing with other teams is often essential to accomplish the goals.

Some positive aspects of communication between inter-teams are people learning new things from other team members, which help to improve their knowledge. Also, TD issues can be found early, if one team gets feedback from others about their work. Our findings also found some issues in inter-team communication which increases the occurrence of TD.

We found TD issues in working with inter-teams in Figure 6 in the results chapter. The issues are:

- Work time might differ for each team, which leads to problems in conducting a meeting, conflict in time between teams and each team should be dependent on other teamwork to complete the product leads to delay in a discussion of TD.
- Conflict in decision making about TD issues. Each team will have their opinions and priorities on a TD, which leads to conflicts between the teams.

Mitigation practices:

- We would suggest that the company accesses different communication channels to support frequent communication between team members.
- A key member of one team should be physically located with other teams to improve communication and to avoid conflicts (Layman et al., 2006).

Intra-team communication

Intra-team communication is communicating with one another inside a team. There are both positive and negative effects in communicating within an intra-team.

It will be easy and convenient to discuss TD related issues within the team because usually, the working time of members within a team will be the same, which leads to the quick arrangement of meetings to discuss TD. The team might not be dependent on other teams' work to accomplish the goal.

Some of the issues are knowledge sharing about TD and its issues will be limited within a team. If the team has communication with other teams, there is a chance to know other issues, which they might not know. It helps to identify and solve TD quickly.

Team distribution

Team co-location tends to improve communication and coordination among team members which has a positive impact on TD. Our findings show that people feel comfortable

discussing related issues face to face in the workplace and it also reduces TD occurrence. On the other hand, the distributed team has a negative impact on TD.
We found TD issues in the distributed team in Figure 7 in the results chapter. The issues are:

- Misunderstandings and confusion due to lack of verbal communication between the team members lead to the occurrence of TD. And if they communicate only via chat, they can likely misinterpret each other's messages.
- Time zone difference and language barriers might cause miscommunication in expressing TD related issues.

Mitigation practices:

- When teams are distributed, always use video conferencing and different modes of communication for discussions and planning. Because it is always easier to discuss things face-to-face and by talking rather than typing (Bless, 2010). It also reduces the chances of miscommunication which helps to reduce the occurrence of TD.
- Synchronizing work hours is one solution suggested in the literature to overcome the lack of face-to-face communication due to time-zone differences (Hossain et al., 2009). This problem can be solved by synchronizing work hours, i.e., by adjusting work hours.
- To avoid language problems, we suggest that the teams encourage people with low language skills to speak slowly. Encouraging team members to speak and asking follow-up questions will improve TD communication by reducing misunderstandings.

Team Size

Our findings show that smaller teams lead to better and easy communication, which in turn helps to manage and reduce TD occurrence.

Additionally, we found that as team size increases, it has both positive and negative impact on TD. On the positive side, the work will get completed soon as there will be more people to take on tasks.

We found TD issues in a large team in Figure 8 in the results chapter. The issues are:

- A new developer joining the team might cause TD.
- The developer in the team might not have some knowledge about the previous work of other developers, which leads to the occurrence of TD.

Mitigation practices:

- Proper and frequent training to new employees should be given regarding the structure and working pattern of the team to avoid the occurrence of TD.
- From Chapter 2.4.6 TD Management, we can say Documentation of TD activities will be helpful to know other people work. From our findings, we found that no proper documentation is maintained for TD. The tools used for documentation are JIRA, which acts as a TD backlog, SonarQube, README files and wiki pages. And the documentation must be updated regularly to avoid the occurrence of documentation debt.

Table 8 gives the description of overall summary of TD impact, advantages, and disadvantages. Here, factors such as team size and team diversity are considered to find the impact, advantages and disadvantages on inter-teams and intra-teams.

Teams	**Team size and diversity**	**Impact on TD**	**Advantages**	**Issues**
Intra-teams	Small teams	Decreases TD	Get to know each other's speciality to discuss TD issues, communication links will be less.	Solving TD issues can take more time if members in a team are less and knowledge sharing will also be limited.
	Large teams	Both increases and decreases TD	Work will be completed soon as each member can pick a task and work.	The developer in the team might not have TD knowledge about previous work and unintentional TD caused by the new developer.
	Co-located	Decreases TD	Comfortable to communicate TD issues directly face to face.	No issues mentioned by interviewees.
	Distributed	Increases TD	No advantages mentioned by interviewees.	Misunderstandings and confusion among team members lead to TD.
Inter-teams	Small teams	Both increases and decreases TD	Easy to communicate and manage TD.	Less knowledge sharing regarding TD activities and their issues.
	Large teams	Both increases and decreases TD	Gain more TD related knowledge and quick feedback from other members.	Difficulty in organizing a meeting and unintentional TD by the new developer.
	Co-located	Both increases and decreases TD	Group meetings where each team can share their TD knowledge and experience.	Conflict in decision making regarding TD and prioritization.
	Distributed	Increases TD	No advantages mentioned by interviewees.	• Miscommunication in expressing TD related issues. • Bias in prioritizing TD • Delay in the discussion of TD issues

Table 8: It gives the description of the overall summary of the impact on TD, advantages and issues based on team size and team diversity in Intra-teams and Inter-teams. Brown colored text represents no advantages/disadvantages.

R.Q 2.2: How does the direct communication of developers with customers have an impact on TD?

We found that collaboration with customers has both positive and negative impacts on TD. On the positive side, developers will get precise requirement details directly from the customer, which can lead to a better understanding of customer needs to develop a final product. Obtaining requirements from other indirect sources might lead to the loss of some important information which might lead to more TD. Missing details of requirements can lead to requirement debt. Requirement debt is leaving comments on features that need to be implemented in the future (Maldonado & Shihab, 2015).

Collaborating with customers in the early phases of a project helps to identify and correct errors and defects in the product and future details of what customer needs helps to improve sprint planning and explaining the issues to customers can reduce TD. Also, developers can get more time to deliver the product if the customer permits.

R.Q 2.3: How does the involvement of developers in decision making have any impact on TD?

Involving developers in decision making has a positive impact on TD. Developers know more about the seriousness of the TD issues rather than a manager or team lead because they are the one who is working with TD. So, suggestions from developers are important to prioritize, for TD to be solved first. From our findings, we found that developers are mostly involved in providing suggestions related to TD issues, the overall decisions were taken by the manager or architect. Some companies' suggestions were taken based on experience i.e., only senior developers are involved in providing suggestions. Junior developers will report their work and issues faced to senior developers. Senior developers might be aware of TD issues faced by junior developers. So, it does not affect the occurrence of TD. However, if the issues are related to cost or reputation decisions were taken directly by the people at the higher level.

Whereas we found that some organizations had a flat hierarchy where the developers had the liberty to come with innovative solutions to TD issues.

6.1.1 Comparing our studies to other studies

Here, the comparison of our studies with other related works is illustrated.

1. (Hummel et al., 2013) considered the Unified Model of SD success to derive the categories to analyze the results of papers. Team size and team distribution are considered as a category in the input dimension that should influence the communication mechanism of Software Development (SD) teams. However, the paper is not focused on TD related communication. Also, this paper has mentioned how scrum activities like a daily standup, sprint planning, retrospective and review have an impact on the communication mechanisms of SD teams. We studied these activities to find how they impacted TD communication.
2. (Melo et al., 2013) discussed inter-team and intra-team have an impact on productivity by considering inter-team coordination and intra-team coordination, which included team member turnover and team design choices. However, the paper is not focused on TD related communication. We have considered this paper to study the impact of TD occurrence by considering inter-team, intra-team, team design choices and team distribution.
3. (Nyrud & Stray, 2017) used framework to map inter-team coordination mechanisms such as Impersonal mode, group mode and personal mode communication in a large-scale program. This framework does not provide details related to TD. So, we used this framework to study in-depth detail about types and modes of communication used to discuss TD.

Table 9 represents the comparison of our findings of factors describing how TD is communicated with works done by (Melo et al., 2013), (Nyrud & Stray, 2017) and (Hummel et al., 2013); along with some additional topics that have an impact on TD.

Studied Case	**(Melo et al., 2013)**	**(Nyrud & Stray, 2017)**	**(Hummel et al., 2013)**
Team size	Team size	Impersonal mode	Team size
Team distribution	Team distribution	Personal mode	Team distribution
Inter-teams	Team member allocation	Group mode- Scrum practices	Group Scrum practices
Intra-teams	Team member skills		Project domain
Types and modes of communication (Scrum practices)	Team member turnover		
Documentation			
Customer collaboration with developers			
Developers' involvement in decision making			

Table 9: It represents the comparison of our findings of factors describing how TD is communicated with works done by (Melo et al., 2013), (Nyrud & Stray, 2017) and (Hummel et al., 2013); along with some additional topics that have an impact on TD.

6.2 Team Organization

Organizing the teams has an important role in the organization's development process, which ultimately helps in achieving qualitative results. This section will have discussions on the following research questions:

R.Q 3: How does organizing a team in a software company affect the occurrence of TD?

The2 team organization plays a significant role in causing TD. The teams are organized in a better way to ensure an effective TD management strategy in the software industry. The effort of teams and their investment in TD-related issues helps them to reduce TD. This can be discussed further and summarized more in detail in RQ 3.1,3.2,3.3, and 3.4.

But in general, we found some issues with the organizing of teams. The issues and the mitigation practices are discussed below from our background studies and our experiences.

- When TD is not discussed during the current planning meetings and keeping it aside or moving it over to upcoming sprints or neglecting them.

Mitigation Practices

- Discuss TD activities during planning sessions and add them to the backlog to make the team members aware of the open TDs in the project.

- Utilization of time wisely in TD activities helps to resolve the TD issues and adequately manage them (Power, 2013).

R.Q 3.1: What are the team organization factors causing TD and how can they be managed?

From Figure 11, we can say that TD arises due to various factors such as employee turnover in the team, inappropriate planning, lack of knowledge, lack of well-defined process, lack of training.

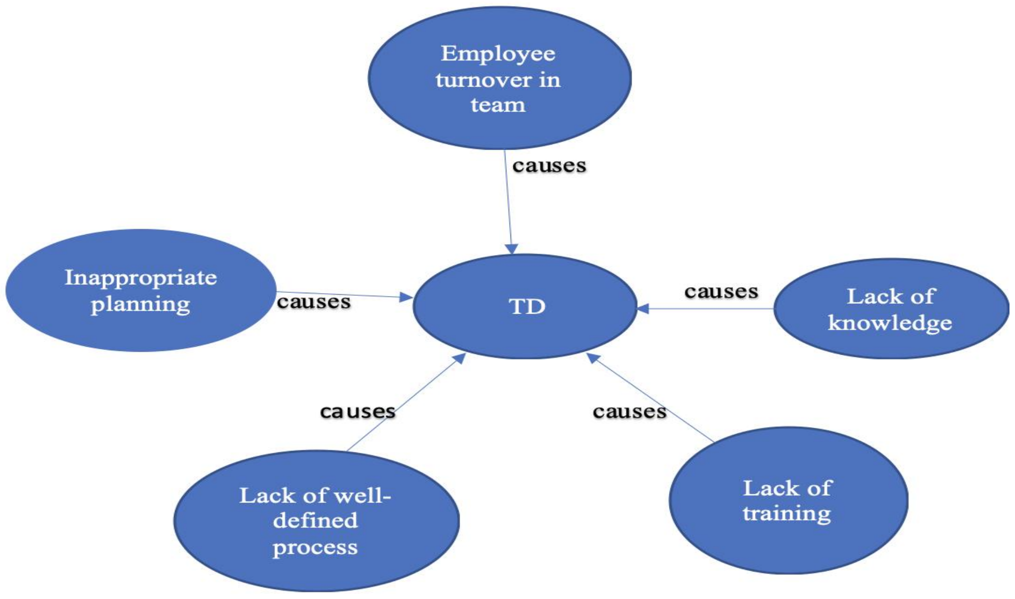


Figure 11: It represents the major factors that cause TD in various organizations. The blue bubble represents each factor of team organization due to which TD occurs in agile teams.

From Table 5 in the results chapter and Figure 11, we found that the main factor which causes TD is employee turnover in the team. When a person leaves the company, and a new person joins instead, it takes some time for the newly joined person to get into the track of work. This may be due to several reasons such as the new tools and technology, the new working culture, and they may not have a good overview of the application.

Some of the management techniques we found are:

- **Proper Documentation**

The progress of the tasks performed by each team member needs to be documented properly which helps in understanding more overview of the application.

- **Conducting Knowledge Transfer (KT) Sessions**

It is a kind of transfer of knowledge from the person who develops the application to the one who is not familiar with the application. It also helps in providing easy and rapid knowledge access to the existing team.

- **Providing Hands-on training**

When persons with no prior experience join an organization, the hands-on training helps them to study and get to know more about the application architecture.

- **Reviewing Code Quality**

The reviewing of codes shares knowledge among the developers which in turn increases the quality of that code.

The next factor which causes TD we found is inappropriate planning. Proper and well-defined planning should be done which involves a description of the activities to be performed on the projects (Freire et al., 2020). If the sprint planning does not involve discussions about TD, the chances to skip them is very high, and more TD may get added to the existing and therefore takes a long time to fix the TD or it sometimes remains unresolved.

Mitigation Practices to improve inappropriate planning

- The team's planning meeting or sprint planning should include TD as a topic so that TD-related activities and their status can be discussed and will reach all team members (Power, 2013).

The next factor we found is lack of knowledge; the developers should possess the skills in terms of domain, technologies, tools, development lifecycle, and knowledge about customer needs, which are required for the development process.

Mitigation Practice to improve lack of knowledge

- From our findings, we would suggest that when a new person joins the organization, the existence of senior members in the team helps to share knowledge and review their work which helps to reduce TD.

The next factor that causes TD that we found is the lack of a well-defined process; describes the steps in the development process. The negligence of developer in properly defining the process lead them to move backwards and perform those again thus leads to TD (Codabux & Williams, 2013)

Mitigation Practice to improve lack of well-defined process

- We suggest that the needs of the customer should be well known to gather the requirements which otherwise divert from the actual wish of the customer and may cause additional work.

The final factor we found as the cause of TD is lack of training; training is relevant for an organization to provide developers with the ease to work with their related domain knowledge, tools, and technologies. We found that most of the organizations do not provide any pieces of training particularly to manage TD. But in few organizations, there is training in general which is mostly voluntary, and in some other organizations, there is not even any kind of training provided.

Mitigation Practice to improve lack of training:

- Developers should be provided with some mandatory training to make them aware of the concept of TD and thus it can reduce TD (Codabux & Williams, 2013).

R.Q 3.2: Does working with common modules have any impact on the occurrence of TD?

From our analysis, we found that all the organizations use common modules or reusable components in one way or another. The commonly used modules are common libraries, reusable code, common infrastructure, common repositories, common guidelines for logging. The reusable components have both positive and negative impacts on TD. But we found that most teams think that the usage of reusable components is helpful for the teams to reduce TD.

Advantages of using reusable components

- Developer time will not be wasted by developing a new function or component every single time.
- It brings sustainability and consistency within and across teams.

Issues

- Since these modules are common, each team using these modules may not have full control over it to optimize.
- It can lead to additional work if they are not maintained properly.

Mitigation Practices

- From our findings, we would say that the modules which are used in common need to be regularly monitored, reviewed, and tested to ensure their quality and accuracy.
- We suggest that the reusable components must be ensured to be up to date and maintained efficiently.

Overall, from our findings in Table 6Table 6: The usage of common modules and their impact on the occurrence of TD. The first column indicates the most common reusable modules used in various organizations; the second column indicates their detailed description, and the third column indicates their impact on the occurrence of TD. and Figure 10: from the results chapter, we found that the usage of common modules in organizations reduces TD.

R.Q 3.3: Is there a need for a separate team and a specific role for managing TD?

From our findings, we found that most of the organizations do not have a separate team and separate roles for managing TD. The developers do not think that the need for a separate team particularly for TD could help them resolve TDs. We found that the development teams themselves are handling the TD, which arises in their teams. Some of the organizations use JIRA where they define or create TD as a task in their backlogs and consider it during planning sessions.

Issue

- When all members in a team jointly focus on TD, it may result in a delay in the development process.

Mitigation Practice

- We would suggest a dedicated member in a team should be assigned to discuss and manage TD-specific tasks so that the other team members could focus on adding new features and functionalities.

R.Q 3.4: What are the impacts of* developers' task distribution *on TD occurrence?

We found that task distribution in organizations is done in two ways: First, the tasks are assigned to developers by the product owners or scrum masters. They make sure that the developers do not work on the same applications every time. From our findings, this task distribution has both positive and negative impacts on TD. The positive side is:

- The developers will have good knowledge of different applications and functionalities and will thus be able to help each other even in TD identification and management.

- The tasks can be allotted to developers based on priority and complexity; thus, high priority tasks are given preference.

The negative side is:

- If the developers have no prior knowledge of the area and domain of the project, they are assigned; there are more chances of generating TD.

Next, tasks are chosen by the developers themselves. We found that in most organizations, developers pick their tasks from a backlog based on their area of interest, domain knowledge, and project experience. We found that this task distribution also has both positive and negative impacts on TD. The positive side is,

- Since the developers choose tasks based on their area of interest, the developers are familiar with the work, which helps them to reduce TD.

On the negative side,

- The developers may not consider the priority of tasks while they choose, thus leaving high priority TDs behind.

6.2.1 Comparing our studies to other studies

Here, the comparison of our studies with other related works is illustrated.

1. (Rios et al., 2018) considered the causes of TD occurrence and the current TD effects. It also concentrates on InsighTD[6] design, which consists of several industrial surveys to investigate the causes and effects of TD. The paper mainly focuses on TD causes such as inappropriate planning, deadlines, lack of knowledge, and lack of a well-defined process. We also considered this paper to analyze the causes of TD and to find the additional factors which cause TD in organizations.

[6]A globally distributed family of industrial surveys on causes and effects of TD. The aim of this design is to analyze the practices and the industrial trends in TD issues such as the causes of TD, its effects and the way software development teams monitor and track TD.

2. (Freire et al., 2020) considered answers from 207 software practitioners and analyzed the actions used to prevent TD in practice. However, in this study, the TD preventive actions in software projects identified are project planning, well-defined requirements, training, adoption of good practices, and creating tests. We studied this paper to know about the factors that cause TD in detail and to understand the impact of best practices on TD occurrence in various organizations.

3. (Codabux & Williams, 2013) focused on studies based on interviews and observations on the implementation of agile practices with an industrial partner within an organization. This paper analyzed the best practices to manage TD, which can be evaluated later by practitioners and researchers to extend the knowledge on TD. We considered this study to assess the impact of training provided to employees in understanding TD-related issues and solving them. We also identified the impacts of task distribution on TD occurrence and analyzed the effect of a dedicated role for managing TD.

Table 10 represents the comparison of our findings of team organization factors and management techniques along with some additional factors causing TD, impact on TD and management techniques.

Studied Case	**(Rios et al., 2018)**	**(Freire et al., 2020)**	**(Codabux & Williams, 2013)**
Employee turnover in team	Deadline	Project Planning	Training
Inappropriate planning	Insufficient knowledge	Well-defined requirements	Work allocation
Lack of well-defined process	Lack of well-defined process	Training	Assign dedicated teams
Lack of knowledge	Inappropriate planning	Creating tests	
Lack of Training		Adoption of good practices	
Dedicated role			
Task Distribution			
Reusable/Common modules			

Table 10: Represents the comparison of our findings of team organization factors and management techniques with works done by (Rios et al., 2018), (Freire et al., 2020) and (Codabux & Williams, 2013) with some additional factors causing TD, impact on TD and management techniques.

6.3 Contributions

We summarized the management of TD that affects team communication and team organization. Some studies explained agile team communication in inter-teams and intra-teams but not related to TD. In this research, we combined both agile team communication and TD to explain communication ways that impact TD. The causes of TD while organizing the teams are also identified and related management techniques have been found and their impact on TD has been analyzed.

Implications for Research

The results of the case study represent a contribution to research.

- Our results are important for future research by providing insights into the investigation of how communication takes place in a team when TD occurs and how it affects TD management.
- We evaluated the most crucial factors of TD in various organizations in detail, which arise because of the organization of teams and focus on various ways to manage them which helps to provide researchers with a better understanding of TD occurrence and its management.

Implications for Practice

The findings of the case study also represent a meaningful contribution to practice.

- Software practitioners can use our findings to understand the impact of TD while communicating in intra-team and inter-team. Also, it helps to choose teams by considering factors like team size and team distribution.
- Choosing the best modes of communication to discuss TD is the key point to solve the issues quickly. Practitioners can refer to this case study while choosing the communication mode.
- Impacts on TD in collaborating with customers are studied and practitioners can use them to reduce the occurrence of TD.
- The main factors causing TD in organizations because of the organization of teams have been identified in this case study and this result can be used to resolve and manage TD more easily.
- The impact of reusable components on TD management is studied and practitioners can use this case study to know how these modules work on TD.
- The developer’s efforts in managing TD must be taken into consideration to rectify it in the future.

6.4 Threats to validity

Validity

The validity of a study denotes the trustworthiness of the results, to what extent the results are true and not biased by the researchers' subjective point of view (Runeson et al., 2012). The types of validity that must be considered in all phases of the study are:

External validity is concerned with to what extent it is possible to generalize the findings and to what extent the findings are of interest to other people outside the investigated case (Runeson et al., 2012). The threat to external validity is the selection of subjects who are irrelevant to the interview. This can be reduced by selecting suitable participants for interviews. We have partially mitigated this by choosing interviewees who only work in an agile process.

Internal validity is concerned with when causal relations are examined (Runeson et al., 2012). It is quite challenging to improve the internal validity in case studies since it may be difficult to know the causalities reasons (Huumo et al., 2016). We conducted semi-structured interviews for this study to achieve in-depth knowledge about the concept of TD and its relationship with team communication and organization factors. The threat in internal validity is that the interviewees might not be much aware of the qualitative factors and might have chances to miss some causalities. This threat can be mitigated by asking questions more precisely to make them aware of the causes and their causalities.

Construct validity is to what extent the operational measures that are studied represent what the researcher has in mind and what is investigated according to the research questions (Runeson et al., 2012). The threat in construct validity is fact that the concepts discussed in the interview questions are not understood in the same way by the researcher and the interviewed persons. The threat can be mitigated by making sure 1) Explaining the purpose of the study to the interviewees before proceeding to the actual interview will help them understand the interview questions and answer them in a better way. 2) Clarifying the purpose of the study before starting the interview will help to obtain more relevant information related to the study. 3) Informing that the interview and the data collection will not reveal the identity of the respondents and consent form agreement is also provided if the respondents demand it. We have followed all the steps mentioned here to mitigate this threat.

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers (Runeson et al., 2012).The threat to reliability is if it is not clear how to code the collected data or if the interview questions are unclear and if another researcher, later on, conducted the same study, the result should be the same. In Chapter 3.3 Case study we mentioned that these threats were mitigated by making sure that 1) The interview instruments were reviewed and evaluated by other researchers. 2) Both researchers were present while interviewing to make sure not to miss the details. 3) Both the researchers were contributing to different phases of data analysis, thus reducing the bias of single researchers. 4) Also, we cautiously explained the process of data collection and analysis to make the study repeatable. We have followed all the steps mentioned here to mitigate this threat.

6.5 Limitations

The limitations of this study include those common to qualitative case study research in general and those specific to this study.

- **Number of respondents:**

The number of respondents in most of the company is low; though all the respondents have relevant knowledge and experience, the findings might have been stronger with more respondents in each company.

- **Use of single data collection method**

We interviewed the participants via video conferencing. If we visited a company and observed, the various methods of data collection can be compared that might lead to more findings.

- **Consideration of different factors**

We have chosen team size and team diversity as the main factors to discuss the occurrence of TD related communication. Also, inter-team and intra-team communication have been considered to know the way that TD is communicated. If other factors are considered like team member seating, team member turnover, the results and discussions could have been different. Also, in team organization, our study deals with the impact of TD during planning sessions but if we consider non-functional testing, the results would have been different.

- **Team members point of view**

The study had various team members' points of view, which mostly includes developers. We had interviewed only one architect and so we assume that the result of the study could have been different if we had interviewed more architect roles to understand the overview of the system.

- **Shortage of Interview Time**

Considering the time factor from the interviewee's point, each interview ranged from 40 to 75 minutes, thus we consider only the main factors of TD, due to organizing teams. If we considered the other factors of TD, we could have gathered some more information and the way it impacts TD.

7 Conclusion and future work

In this thesis, we have presented a case study on team communication and team organization in agile teams and identified its impact on TD management. Data were collected in our study by conducting 14 interviews from ten different organizations of different roles.

The first research questions aimed at understanding how agile teams communicate when TD occurs. We found that teams use group communication such as daily standup, sprint planning and refinement meetings to discuss TD issues. Also, teams use communication channels such as JIRA, slack, sonarQube and kanban board to communicate TD when it occurs.

The second research question explains how team communication affects the occurrences of TD. We found that small and co-located teams are preferred more by teams to communicate TD. Intra-team communication is preferred by the teams to communicate TD. Positive effects of using intra-team communication are that it will be easy and quick to arrange a meeting to discuss TD and less dependent on other teamwork, whereas inter-team communication has both positive and negative impacts on TD. One of the positive aspects is that team members can learn new things about TD from other teams and the negative aspects are there will be a conflict in decision making about TD issues. We have also found both positive and negative impacts on TD when there is a close collaboration with customers. Positive impact includes that developers can get precise requirement details and negative impact is missing details of requirement can lead to requirement debt. Next, there is a positive impact on TD when developers are involved in decision making.

The last research question aimed at understanding the main factors causing TD when organizing teams and the different ways it can be managed. We found that the main factor of TD occurs due to employee turnover in the team followed by inappropriate planning, lack of knowledge, lack of well-defined process and finally lack of training. We discussed the need for a separate role or dedicated team member particularly working on TD related activities. When TDs are considered and discussed during sprint planning, every team member becomes aware of TD activities in their team, which helps them to work on and manage TD. There is a decrease in TD when tasks are picked by developers rather than being assigned to them.

Finally, we observed that TD decreases when organizations use common modules such as common libraries, infrastructure codebase, repositories, guidelines for logging, building blocks for applications.

Overall, from our case study, we found the communication tools used by software teams when TD occurs and analyzed the major team communication and team organization factors causing TD. Also, their impact on TD occurrence has been identified. This study also provides some management techniques, which help to monitor and manage TD.

7.1 Future work

Several topics could be interesting to research further.

- Conducting more case studies on this topic to further validate and verify the findings which help to supplement the research.
- In this study, we mainly interviewed roles such as Project Manager, Tech leads, Scrum master/Business analyst, Architect, and few developers in most of the companies. It would be interesting to study a few more roles like testers in each company to obtain their experience in handling TD and to involve more architect roles to get the overview of the system which helps to better understand the impact of TD on different teams.
- The identification of TD types in various organizations and the risks associated with it can give more specific details about how types of TD are communicated.
- Lastly, this thesis is focused on managing TD that affects team communication and team organization of agile teams, which are considered only two among the factors of TD, but it would be good for research to concentrate on the other factors in agile teams. For example, how communication of TD in an agile team has an impact on team performance and productivity and to analyze how teams can be effectively and efficiently formed or organized, which affects the performance of the teams and its impact on TD.

References

Agerfalk, P. J., Fitzgerald, B., Holmstrom Olsson, H., Lings, B., Lundell, B., & Ó Conchúir, E. (2005). A framework for considering opportunities and threats in distributed software development.

Alfayez, R., Alwehaibi, W., Winn, R., Venson, E., & Boehm, B. (2020). A systematic literature review of technical debt prioritization. IEEE/ACM Proceedings of the 3rd International Conference on Technical Debt,

Alves, N. S., Ribeiro, L. F., Caires, V., Mendes, T. S., & Spínola, R. O. (2014). Towards an ontology of terms on technical debt. 2014 Sixth International Workshop on Managing Technical Debt,

Ampatzoglou, A., Mittas, N., Tsintzira, A.-A., Ampatzoglou, A., Arvanitou, E.-M., Chatzigeorgiou, A., Avgeriou, P., & Angelis, L. (2020). Exploring the Relation between Technical Debt Principal and Interest: An Empirical Approach. *Information and Software Technology*, *128*, 106391.

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

Basili, V. R., & Weiss, D. M. (1984). A methodology for collecting valid software engineering data. *IEEE Transactions on software engineering*, *10*(6), 728-738.

Bavani, R. (2012). Distributed agile, agile testing, and technical debt. *IEEE software*, *29*(6), 28-33.

Besker, T., Martini, A., Lokuge, R. E., Blincoe, K., & Bosch, J. (2018). Embracing technical debt, from a startup company perspective. 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME),

Bill, C., Sappidi, J., & Szynkarski, A. (2012). Estimating the size, cost, and types of technical debt. 2012 Third International Workshop on Managing Technical Debt (MTD),

Bless, M. (2010). Distributed meetings in distributed teams. n International Conference on Agile Software Development, Berlin, Heidelberg.

Bricki, N., & Green, J. (2007). *A guide to using qualitative research methodology*.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., Sangwan, R., Seaman, C., Sullivan, K., & Zazworka, N. (2010).

Managing technical debt in software-reliant systems. Proceedings of the FSE/SDP workshop on Future of software engineering research,

Chau, T., & Frank, M. (2004). Knowledge Sharing in Agile Software Teams. In *In Logic versus approximation* (pp. 173-183). Springer.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer, 34*, 131-133.

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

Codabux, Z., Williams, B. J., Bradshaw, G. L., & Cantor, M. (2017). An empirical assessment of technical debt practices in industry. *Journal of software: Evolution and Process*, *29*(10), e1894.

Codabux, Z., Williams, B. J., & Niu, N. (2014). A quality assurance approach to technical debt. Proceedings of the International Conference on Software Engineering Research and Practice (SERP),

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches (3rd edition)*. SAGE.

Creswell, J. W., & Creswell, J. D. (2018). *Research design: qualitative, quantitative, and mixed methods approaches*. SAGE.

Dalton, D. R., & Todor, W. D. (1979). Turnover turned over: An expanded and positive perspective. *Academy of management review*, *4*(2), 225-235.

Dingsøyr, T., & Brede, N. M. (2014). Towards Principles of Large-Scale Agile Development. nternational Conference on Agile Software Development XP 2014: Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation,

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. In: Elsevier.

Farid, W. M. (2012). The Normap methodology: Lightweight engineering of non-functional requirements for agile processes. 2012 19th Asia-Pacific Software Engineering Conference,

Freire, S., Rios, N., Mendonça, M., Falessi, D., Seaman, C., Izurieta, C., & Spínola, R. O. (2020). Actions and impediments for technical debt prevention: results from a global family of industrial surveys. Proceedings of the 35th Annual ACM Symposium on Applied Computing,

Fritzsche, M., & Keil, P. (2007). Agile methods and CMMI: compatibility or conflict? *e-Informatica Software Engineering Journal*, *1*(1).

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

Gonçalves, L. (2018). Scrum. *Controlling & Management Review,, 62(4),* 40-42.

Gren, L., Goldman, A., & Jacobsson, C. (2019). The perceived effects of group developmental psychology training on agile software development teams. *arXiv preprint arXiv:1911.09057*.

Guo, Y., & Seaman, C. (2011). A portfolio approach to technical debt management. Proceedings of the 2nd Workshop on Managing Technical Debt,

Guo, Y., Spínola, R. O., & Seaman , C. (2016). Exploring the costs of technical debt management – a case study. *Empir Software Eng 21*, 159-182.

Hayat, F., Rehman, A. U., Arif, K. S., Wahab, K., & Abbas, M. (2019). The Influence of Agile Methodology (Scrum) on Software Project Management. 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD),

Highsmith, J., & Cockburn, A. (2001). Agile Software Development: The Business of Innovation. *Computer*, *34*, 120-122.

Hoda, R., & Murugesan, L. K. (2016). Multi-level agile project management challenges: A self-organizing team perspective. *Journal of Systems and Software*, *117*, 245-257.

Hossain, E., Babar, M. A., & Verner, J. (2009). Towards a framework for using agile approaches in global software development. In International Conference on Product-Focused Software Process Improvement, Berlin, Heidelberg.

Hummel, M., Rosenkranz, C., & Holten, R. (2013). The Role of Communication in Agile Systems Development. *Business & Information Systems Engineering*, 343-355.

Huumo, J. Y., Maglyas, A., & Smolander, K. (2016). How do software development teams manage technical debt? –An empirical study. *The Journal of Systems and Software 120*, 195-218.

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

Layman, L., Williams, L., Damian, D., & Bures, H. (2006). Essential communication practices for Extreme Programming in a global software development team. *Information and software technology,, 48*(9), 781-794.

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, *43*, 59-67.

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.

Mahalakshmi, M., & Sundararajan, M. (2013). Traditional SDLC vs scrum methodology--a comparative study. *International Journal of Emerging Technology and Advanced Engineering*, *3*, 192-196.

Mäntylä, M. V., & Lassenius, C. (2008). What types of defects are really discovered in code reviews? *IEEE Transactions on software engineering, 35*(3), 430-448.

Martin, R. C. (2002). *Agile software development: principles, patterns, and practices*. Prentice Hall.

Martini, A., & Bosch, J. (2015). Towards prioritizing architecture technical debt: information needs of architects and product owners. 2015 41St euromicro conference on software engineering and advanced applications,

Martini, A., Bosch, J., & Chaudron, M. (2015). Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study. *Information and Software Technology*, *67*, 237-253.

Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015). Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. *SIGSOFT Softw. Eng. Notes*, 1-6.

McDowell, S., & Dourambeis, N. (2007). British Telecom experience report: agile intervention–BT's joining the dots events for organizational change. International Conference on Extreme Programming and Agile Processes in Software Engineering, Melo, C., Cruzes, D., Kon, F., & Conradi, R. (2013). Interpretative case studies on agile team productivity and management. *Inf. Softw. Technol*, 412-427.

Morris, D. (2017). *Scrum in easy steps: An ideal framework for agile projects*. In Easy Steps Limited.

Nyrud, H., & Stray, V. (2017). Inter-team coordination mechanisms in large-scale agile. XP '17: Proceedings of the XP2017 Scientific Workshops,

Ozer, M., & Vogel, D. (2015). Contextualized relationship between knowledge sharing and performance in software development. *Journal of Management Information Systems*, *32*(2), 134-161.

Perry, D. E., Sim, S. E., & Easterbrook, S. M. (2004). Case studies for software engineers. Proceedings. 26th International Conference on Software Engineering,

Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering*, *15*(6), 654-693.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still , J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering,*, *13*(3), 303-337.

Power, K. (2013). Understanding the impact of technical debt on the capacity and velocity of teams and organizations. In IEEE International Workshop on Managing Technical Debt (MTD),, San Francisco.

Ribeiro, L. F., F. Farias, M. A. d., Mendonça, M., & Spínola, R. O. (2016). Decision Criteria for the Payment of Technical Debt in Software Projects: A Systematic Mapping Study.

Rios, N., deMendonça Neto, M. G., & Spínola, R. O. (2018). A tertiary study on technical debt: Types, management strategies, research trends, and base information for practitioners. *Information and Software Technology*, *102,* 117-145.

Rios, N., Spínola, R. O., Mendonça, M., & Seaman , C. (2020). The practitioners' point of view on the concept of technical debt and its causes and consequences: a design for a global family of industrial surveys and its first results from Brazil. *Empir Software Eng 25*, 3216–3287.

Rolland, K.-H., & Lyytinen, K. (2021). Exploring the Tensions Between Management of Architectural Debt and Digital Innovation: The Case of a Financial Organization. Proceedings of the 54th Hawaii International Conference on System Sciences,

Rubin, K. S. (2012). *Rubin, K. S. (2012). Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley.

Runeson, P., Host, M., Rainer, A., & Regnell, B. (2012). Case study research in software engineering: Guidelines and examples. In. John Wiley & Sons.

Sharp, H., Robinson, H., & Petre, M. (2009). The role of physical artefacts in agile software development: Two complementary perspectives. *Interacting with computers*, *21*(1-2), 108-116.

Shilpa, B., & Ingle, M. (2010). Analyzing the modes of communication in agile practices. 2010 3rd International Conference on Computer Science and Information Technology,

Shrivastava, S. V., & Date, H. (2010). Distributed Agile Software Development: . *JOURNAL OF COMPUTER SCIENCE AND ENGINEERING, 1*(1), 10-17.

Soares, H. F., Alves, N. S., Mendes, T. S., Mendonça, M., & Spinola, R. O. (2015). Investigating the link between user stories and documentation debt on software projects. 2015 12th International Conference on Information Technology-New Generations,

Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. 2017 International Conference on Computing, Communication and Automation (ICCCA),

Stray, V., Moe, N. B., & Bergersen, G. R. (2017). Are Daily Stand-up Meetings Valuable? A Survey of Developers in Software Teams. International Conference on Agile Software Development,

Stray, V. G., Moe, N. B., & Dingsøyr, T. (2011). Challenges to teamwork: a multiple case study of two agile teams. International conference on agile software development,

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

Yli-Hummo, 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. International Conference of Software Business ICSOB 2015,

Zia, A., Arshad, W., & Mahmood, W. (2018). Preference in using agile development with larger team size. *International Journal of Advanced Computer Science and Applications*, *9*(7), 116-123.

Appendices

A. Interview Guide

Introduction

- Present ourselves and say a little about the project.
- Thank the person for participating.
- Confirm confidentiality and anonymity.
- Ask for permission to record the interview.

General

- Can you tell us about yourself?
- How long have you been working for this company?

Team

- How is your team composed of?
- How many members are there in your team?

Communication

- How do you communicate factors of TD within the team?
- Which teams do you think will be easy to communicate these factors?
- Which team (co-located or distributed team) is easy to communicate TD? Why?
- Can you tell us some advantages and issues of TD communication in working with intra-team?
- Which type of communication do you prefer to discuss the factors of TD?
- Personal communication or group communication?
- Who do you talk to if you find TD?
- Does close collaboration with customers have an impact on TD?

Organization

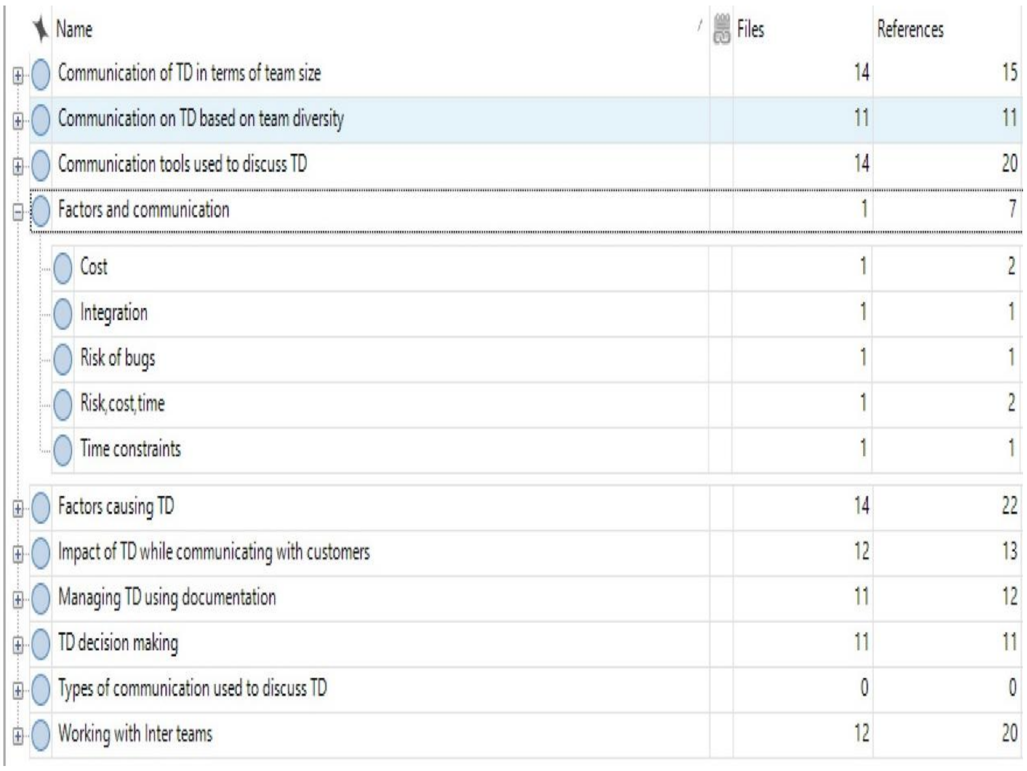
- Does TD arise in your team because of any of the factors?
- Does the planning session include discussions on TD?
- How are tasks distributed to team members and does it help to manage TD?
- Does frequent change in team members affect TD management?
- Do you experience working with common modules in your team and how does it affect TD?

Closing

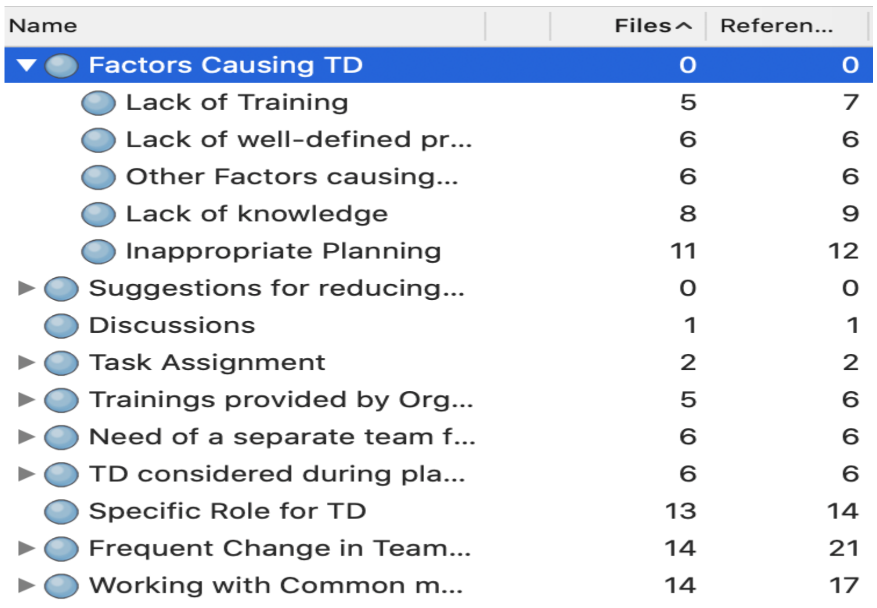
- Are there any personal suggestions you would like to add to reduce or manage TD?
- Thank you for participating!

B. NVivo Coding

The below two picture represents the codes for team communication and team organization performed in NVivo.



Name	Files	References
Communication of TD in terms of team size	14	15
Communication on TD based on team diversity	11	11
Communication tools used to discuss TD	14	20
Factors and communication	1	7
Cost	1	2
Integration	1	1
Risk of bugs	1	1
Risk,cost,time	1	2
Time constraints	1	1
Factors causing TD	14	22
Impact of TD while communicating with customers	12	13
Managing TD using documentation	11	12
TD decision making	11	11
Types of communication used to discuss TD	0	0
Working with Inter teams	12	20



Name	Files	Referen...
Factors Causing TD	0	0
Lack of Training	5	7
Lack of well-defined pr...	6	6
Other Factors causing...	6	6
Lack of knowledge	8	9
Inappropriate Planning	11	12
Suggestions for reducing...	0	0
Discussions	1	1
Task Assignment	2	2
Trainings provided by Org...	5	6
Need of a separate team f...	6	6
TD considered during pla...	6	6
Specific Role for TD	13	14
Frequent Change in Team...	14	21
Working with Common m...	14	17