Responding to Changes: A Grounded Theory of Agile Activities in Software Development

Master's Thesis

Supervisor: Jacob Nørbjerg

Department of Digitalization

Written by: Jun Sun (150297), CMIBO1000E, M.Sc. Economics & Business Administration (MIB)

Number of characters (incl. spaces): 101,476

Number of standard pages: 58

Submitted: 15. May 2023

-This page is intentionally left blank-

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Jacob Nørbjerg, for sharing your wealth of knowledge on the topic and guiding me through the research process. Your unwavering support has enabled me to use my capabilities fully, and I have gained invaluable insights from your mentorship.

Secondly, I would like to thank Steffen Gabriel Brandenhoff for generously sharing your extensive network and knowledge about agile software development. Your support has been invaluable in shaping the course of my research.

I also want to express my appreciation to the six informants who provided me with extensive and insightful feedback. Your willingness to share your time and expertise has contributed significantly to the success of this project.

Lastly, I am grateful to my family members for their unwavering support throughout my journey toward completing my master's degree. Your encouragement and love have been a constant source of motivation.

Abstract

This paper aims to address the challenges of responding to changes in small agile teams by presenting "a grounded theory of agile activities" - a high-level descriptive theory that explains how these teams manage their activities to respond to changes in software development field. The thesis identifies four main domains of agile acts - communication and collaboration, learning, balancing, and adapting - that help agile teams develop the necessary abilities to respond to changes. Additionally, the thesis highlights how these four domains interplay with each other to build an adaptive and evolvable system that facilitates agile teams. Furthermore, this thesis provides managerial implications for both agile and management teams to navigate the volatile, uncertain, complex, and ambiguous (VUCA) environment more effectively.

Keywords: Agile software development; Agile activity; Grounded theory; Small Agile teams; Responding to changes

List of Figures

List of Tables

Table of Contents

1. INTRODUCTION

The information age has led to a plethora of changes and innovations in software development and related fields. Today, software is ubiquitous, embedded in non-technology market offerings such as watches, door locks, toilets, soda fountains, vacuum cleaners, and even light bulbs (Andreessen, 2011). Due to the rapid changes in technology and platform terrains, software teams are under immense pressure to quickly adapt to the latest trends (Hoda et al., 2013). To succeed in this dynamic environment, software teams must possess a diverse range of skills and abilities.

Agile methodologies are designed to be responsive to innovation, creating new knowledge that delivers value to businesses and allowing teams to rapidly respond to competitive challenges (Highsmith, 2002).Highsmith & Cockburn (2001) posit that we must confront the business need for relentless innovation and forge the future work-force culture to thrive in a turbulent environment. Hence, many researchers and practitioners believe that agile methodology is a natural fit for software development. Innovation in the market which requires agile software development to seize or innovation of agile software development method itself, are equally important for people to embrace the changes and challenges of society.

In addition, the process of innovation and change is highly dependent on organizations, cultures, and needs. The actions and preferences of builders, operators, and end-users can significantly influence the process from beginning to end (Mergel et al., 2021). Therefore, it is essential to understand the fundamental core of agile software development methodology before adopting any specific practices. According to S. Kiv et al. (2018), the best way to achieve this is to understand the principles of the Agile Manifesto. Similar to the architecture in software development, which facilitates system integration and coordination among different parts of the system (Agarwal & Tiwana, 2015), the Agile Manifesto guides individuals and organizations to evolve and respond to changes effectively.

As few studies have examined the ways in which agile teams respond to changes in software development on a day-to-day basis, and there is a lack of established theory explaining how agile practices are executed in response to changes, this thesis aims to answer the research question: *"How do small agile teams respond to changes in software development?".* Conboy's (2009) focus group study on how agile practices influence customer value revealed that the same practice is perceived differently across teams, highlighting the importance of considering the project or organizational context when analyzing the research question.

Given the unique context of software development, the researcher will utilize the grounded theory method, a qualitative research methodology that systematically collects and analyzes data to generate theories or explanations, to understand team members' emerging concerns and their responses or solutions to the problems. By doing so, the researcher hopes to initiate a conversation on the main components of agile activities applied by agile teams and how the interrelations among these activities play a role in responding to changes.

After collecting empirical data from agile practitioners and analyzing it using the grounded theory method, the researcher identified four categories of influential enablers for agile teams to respond to changes: communication and collaboration acts, learning acts, balancing acts, and adapting acts. Additionally, the researcher developed a theory with a systematic view that links these enablers to build an evolving software development system, which can help agile teams better adapt to constant changes and contribute more to innovations.

2. CONTEXT & LITERATURE REVIEW

2.1 CONTEXT AND DEFINITION

As this research seeks to understand how agile teams respond to changes in software development, it is important to delimit and conceptualize the key concept the research will build on: the agile history, agile value, agile teams, and the agile methodology in software development.

2.1.1 The History of Agile

Before the agile method was finally developed and implemented into the software industry, there were other adaptations following the waterfall process. For example, the incremental and iterative techniques focusing on breaking the development cycle into pieces evolved from the Waterfall model (Beck, 1999). Incremental development aimed to reduce development time by breaking the project into overlapping increments. Evolutionary methods like iterative development and the Spiral Model (Boehm, 1988) aimed to better handle changing requirements and manage risk. However, Agile practitioners found that a process needs to accept changes rather than stress predictability to be truly agile (Schwaber, 2002). They came to realize that methods that would respond to change as quickly as it arose were necessary (Turk et al., 2002). New methodologies emerged at that period to respond to the agile trend, for instance extreme programing, Crystal, SCRUM, Feature driven development, and others.

In general, Agile methods are a reaction to a traditional way of and acknowledge the need for an alternative to documentation-driven and heavy-weight software development process (Cohen et al., 2004). Most of the Agile practices are nothing new (Cockburn & Highsmith, 2001). The focus and values behind Agile Methods differentiate them from more traditional methods (Cohen et al., 2004). This research defined agile methods based on the previous concepts, and

proposed that agile methods are the ones that embrace changes and keep it as a frame of mind, since [some] processes may look Agile, but they won't feel Agile (Cockburn, 2002).

2.1.2 The Definition of Agile Values

To understand agile methodologies, we need to know the core values from the beginning. In 2001, seventeen agile practitioners came together and proposed an Agile manifesto, including four agile values (Beck et al., 2001):

- *Individuals and interaction over process and tools,*
- *Working software over comprehensive documentation,*
- *Customer collaboration over contract negotiation,*
- *Responding to change over following a plan.*

That is, while there is a value in the items on the right, we value the items on the left more.

Conboy (2009) defines Agility as a software development team's ability to create change, respond to change, and learn from change to deliver value." Adolph (2006) used a similar definition to avoid specifying methodologies and practices. He referred to it simply as the team's ability to deliver value more quickly by responding to change and improving its processes. The core agile value, hereafter, can be concluded as how the team responds to changes to deliver more value. Thus, the fourth Agile value, responding to change over the following plan, is crucial to understanding the essence of Agility.

Furthermore, no matter what kind of agile values or agile methods, are all needed to be executed by people, namely, the agile team. As (Moe et al., 2009) explained, a software development team depends significantly on team performance, as does any process that involves human interactions. Thus, in the following section, the researcher will define the agile team to understand its core characteristics.

2.1.3 The Definition of Agile Team

According to Katzenbach & Smith (1993), a common definition of a team is a small group of people with complementary skills committed to a common purpose, set of performance goals, and approach for which they hold themselves mutually accountable. Self-organizing Agile teams are composed of individuals [that] manage their own workload, shift work among themselves based on need and best fit, and participate in team decision-making (Highsmith, 2004). The software development team (for instance, a Scrum team) is a group of people with different skills, such as UX, testers, product designers, scrum masters, and developers. Moreover, the team keeps the same goal to deliver value in an iterative and fast way. The agile team has autonomy, but to what extent can autonomy be granted and accepted by individuals or the organization based on the agile maturity level and the phase of agile transition (Hoda & Noble, 2017). Furthermore, self-organizing team or individual is the core stone of taking more initiatives regarding self-learning and reflective practices. Only the team or individual can decide on their own work, self-learning and reflection will emerge (See Figure 1). Thus, in this thesis, we define the agile team in software development as a small group of people with complementary skills and commit to the same goal with self-managing autonomy.

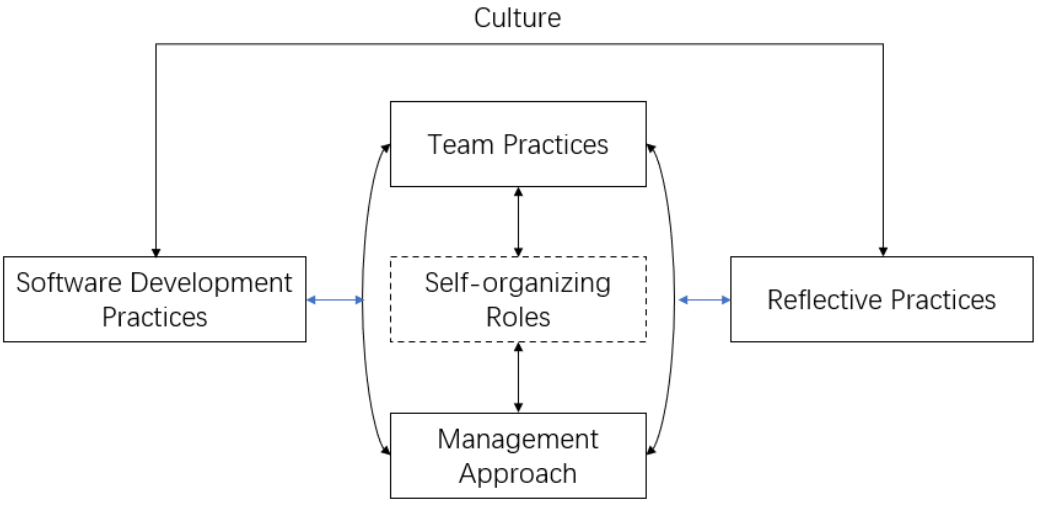

Figure 1. The interactions among Self-organizing Roles and other ISD elements. Adopted from Hoda & Noble (2017)

2.1.4 The Definition of Agile Software Development Methods

Since the main topic of this thesis is about agile activities, it is necessary to understand the different types and characterizations of agile software development methods before we delve into the detailed activities. According to Abrahamsson et al. (2002), Agile software development methods, "officially" started with the publication of the agile manifesto. They defined agile software development method as the one that is incremental, cooperative, straightforward (the method itself is easy to learn and modify) and adaptive (able to make last moment changes) after analyzing the main methods, such as XP, Scrum, Crystal, Feature-driven development, and the others.

Although some agile software development methods have faded out due to their maneuverability and effectiveness, the core characteristics remained: incremental, iterative, cooperative, adaptive, and others, deriving from the agile manifesto. Agile software development methods provide a novel way of approaching software engineering problems, while also maintaining that the methods are by no means exhaustive or capable of solving all the problems (Abrahamsson et al., 2002). Thus, the researcher defined the agile software development methods in this thesis that the ones with the core characteristics of agile values instead of specifying the concrete methods to capture the values most from agile activities.

2.2 LITERATURE REVIEW

2.2.1 Agile Software Development Prior Research

Introductions to and overviews of agile software development have been studied by many researchers, among which Abrahamsson et al. (2002), Cohen et al. (2004), Erickson et al. (2005), and Dybå & Dingsøyr (2008) described various agile methods and lessons learned from applying techniques in the industry. According to Dybå & Dingsøyr (2008), agile studies are grouped into four themes: Introduction and adaptation, Human and social factors, Perceptions on agile methods, and Comparative studies. Dybå & Dingsøyr (2008) analyzed 36 articles (of 1996

articles), and 78% of the agile methodology discussed was XP, which is no longer a primary methodology in software development. Thus, Dybå & Dingsøyr (2008) suggested that further studies on other agile methods, such as Scrum and Lean software development, would be an optimal choice. Due to the rapid increase in researchers' interest in agile software development, the requirement to define Agile comprehensively is in demand.

Conboy (2009) indicates that there are a number of significant conceptual shortcomings with agile methods and the associated literature in its current state, including a lack of clarity, theoretical glue, parsimony, limited applicability, and naivety regarding the evolution of the concept of agility in fields outside systems development. Thus, Conboy (2009) develops a definition and formative taxonomy of agile in an information software development (ISD) context: agility is the continual readiness of an ISD method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment. Dingsøyr et al. (2012) released a paper about the overview of agile methodologies in a decade, from 2003 to 2011. The paper provides an overview of particular issues and a section on agile software development, including 32 articles. Some new perspectives are mentioned, for example, introducing agile processes into an organization working in an ISO 9000 (Williams & Cockburn, 2003), which has never been discussed, becoming a start of a trend of implementing agile methodologies in different industries.

Gradually, researchers started to observe agile software development through different lenses. Some of them focus on balancing perspective to discuss the difference components in agile software development and how the agile teams respond to changes through balancing activities. For example, Ågerfalk et al. (2009) discussed how to balance flexibility and control, and the challenges of agile in distributed projects. Lindskog & Netz (2021) argue how to balance the stability and change when the software development project implementing agile methodology.

Moreover, some researchers discuss how management support influences the agile teams' capabilities and the way to improve them. Ngwenyama & Nørbjerg (2010) investigated software process improvement with weak management support and identified key features of intra-organizational alliances for change management for software process improvement and their relations. Senapathi & Srinivasan (2012) developed a model to explain post-adoptive usage of agile practices, which indicates that top management support is one of the key factors. The influence of the management team is critical to discuss since, as the main project's external context, the management team's support enhances or hinders agile teams from developing adaptive and emergent culture which contribute to the team's agile maturity.

Recently, some researchers have started to link top-down view – management strategy to bottom-up view – project or team activities to build hypothesis based on complex adaptive system, discussing relationship between agile capabilities and emerging strategy initiatives and eventually project success. As posit by (Kaufmann et al., 2020), autonomous, customer-focused teams more often raise their voice and bring in new opportunities and threats. On the other hand, upper managers, are more open to listen to these voices, discuss their implications, and take charge for new strategic initiatives. It becomes more likely that new strategic paths may emerge.

Overall, agile software development research has developed through long period of time, utilizing diversified methods, quantitative and qualitative, to build theories, propositions, and implications, which are helpful for agile practitioners to form their own agile methodologies.

2.2.2 Agile values and Agile practices in Software Development

To understand how agile teams respond to the changes in software development, we need to review relevant literature about Agile values and Agile practices, since responding to changes is one of essential value of Agile manifesto. The fundamental agile values have been introduced in the first section of this part. However, Agile values have been evolving since 2001. Hohl et al. (2018) interviewed the originators of the agile manifesto to see what they retrospect about the agile values and principles and what they think will be the future trend. The informative

conclusions emphasize the importance of considering the variety of different practices and methods that influenced the development of the manifesto (Hohl et al., 2018). Furthermore, the originators of the agile manifesto mentioned that people should question their current understanding of "agile" and recommend reconsidering the core ideas of the manifesto. Hohl et al. (2018) present a future research direction to investigate various agile practices and methods that had an influence on the manifesto, reflecting on this thesis, seeing how agile teams taking daily agile activities to respond to changes.

Meanwhile, according to Madi et al. (2011), agile values are categorized into 17 from the agile manifesto.org website. They surveyed to identify the most critical values to the organizations. Madi et al. (2011) concluded that the values under the people dimension are more critical than the process and product (technology) dimensions, while the values under the process dimension come in the second level of importance. The paper lists the top ten crucial agile values without linking them to detailed agile activities. As the article suggested, further research is required to identify the most effective agile practices under each agile value determined from the analysis, which is the topic of our study. Conboy (2009) proposed a taxonomy of agility to identify agile activities and argued that even though some activities are not commercially labeled or publicly accepted as agile, they can be considered agile activities if they comply with the agile taxonomy. Conboy's argument broadens the extent of agile activities and increases the research flexibility for agile software development since agile activities in specific contexts can be considered.

2.2.3 Responding to Changes in Agile Software Development

As discussed in the introduction part, the future of our Information Age belongs to the individuals and organizations that can respond to changes. Taking the referential data from Forbes 500, Highsmith (2002) measures the increasing change and turbulence in the attrition rate of companies in the past 25 years. He found that the attrition rate increased from 10% to 36%; namely, more companies dropped off from the list now than before. Nevertheless, while everyone admits to the increased pace of change, far fewer understand its implications

(Highsmith, 2002), not to mention how to respond to them. The following literature reviews will discuss some key domains that accommodate responding to changes.

Communication & Collaboration in responding to changes

Denning (2013) elaborates companies' failure in competition, due to neglecting continuous product and process innovation based on insights gained from interacting with their customers. Denning (2013) believes that the focus on value creation should be transformed from internal (cost-saving) to external (customer collaboration), which is similar to the argument of Hoda et al. (2010), who posit that adequate customer involvement and collaboration is one of the most essential influencers that Agile teams need to focus. Hoda et al. (2010) propose the Agile Undercover - a set of strategies the teams use to practice Agile despite insufficient or ineffective customer involvement to mitigate the negative influence. Changing customers' mindsets, changing priority, delegating customer proxy, and others are all strategies to improve the communication and collaboration between agile teams and customers (Hoda et al., 2010).

Wufka and Ralph (2015) propose a preliminary process theory to explain how the interplay between stakeholders (developers, management and users) drives the need for changes; the iterations between recognition and response produce the changes; and the changes take place within the interactions between the team, the development process, and the software artifacts. Agile organizations create chaos for their competitors, first by creating change so fast that competitors are left gasping for breath; and second, by responding quickly to competitors' attempts to change the market (Highsmith, 2002). But how well the process theory of agility explains the practical experiences of industrial agile practitioners remains to be studied (Hoda & Noble, 2017). This thesis will identify how agile teams perform in their daily activities when responding to changes and to develop the theory of agility in software development.

Balancing acts in responding to changes

Some researchers take further steps to analyze practical experiences regarding balancing acts in agile software development. Hoda et al. (2013) discuss iteration pressure and self-learning, which is one of the main supporting activities to responding to changes. They found that the practices most closely related to learning were also those most often sacrificed or poorly implemented over time. The paradox is that continuous learning is essential to maintaining a team that can continue its effectiveness (Hoda et al., 2013). They posit practical strategies for achieving a balance between iteration pressure and continuous learning through expectation management, reflective practice, retrospective, and learning spik. Hoda et al. (2013) developed a new terminology: “learning debt”, inspired by “technical debt” in software development to raise the attention from agile practitioners to balance learning tensions.

The time pressure of agile practices not only exists in the self-learning domain, but also in the field of how agile teams make the proper decision in short period of time. Drury-Grogan et al. (2017) identify specific characteristics of decision making in agile software development, stating that decision in fast agile software development process always have short-term focus, and decisions on complex issues always deferred. According to Drury-Grogan et al. (2013)’s research, the decisions of complex issues in agile software development are usually procrastinated, causing the similar issue as Hoda et al. (2013) posit “Learning debt”, but here can be named as the “decision debt”. To tackle the issue, the agile practitioners need to balance the time pressure by reflecting, learning, and adapting, the same ways as how the agile teams address the technology debt, trying figure out the proper process, time allocation and others to maintain the “debt” in acceptable threshold.

However, on the contrary, time pressure (time box) provides a good sense to the agile teams to focus on the temporary, urgent, and most valuable items so that they can get the quick response out of the short period of time instead of spending great amount of time in discussing complex issues, and perfecting the products without final decisions and outputs.

Moreover, some balancing acts emerge between stability and change in agile teams. Lindskog & Netz (2021) use the grounded theory approach to explore what happens in practice when software teams implement and work Agile, figuring out several inherent processes are ongoing simultaneously to balance the need to have both stability and change. Stability is consistent in agile software development since agile needs some foundation to support the change, such as the Scrum facilitating structure, Sprint planning, Daily standup meeting and so on. When discussing balancing stability and change, we need to be context-oriented and with dynamic thinking, identifying when and on which occasion we need certain stability so that the agile teams can gain more from changes. Balancing acts help the agile teams to develop from "being Agile" to "doing Agile" (Lindskog & Netz, 2021; Denning, 2016a).

Cost of responding to changes

Although responding to change is good as it helps the agile teams to seize the valuable feedback from customers and improve the working efficiency, the adverse side of responding to changes is always existing. In the waterfall model context, Boehm (1976) summarized the cost-of-change curve, finding that a defect in different phases costs differently; in the earlier phase, the cost is lower and vice versa. Nevertheless, as the rising number of changes are result of an external change from the environment—from customers, competitors, and governments—then whether it is cheaper to catch "defects" upfront is irrelevant (J. A. Highsmith, 2002). The right strategy, according to Agilists, is to work on reducing the cost of change throughout the project by developing simple solutions (less to change, easier to change), refactoring (making the design easier to change), and constant testing (early, less expensive, defect reduction) (J. A. Highsmith, 2002).

Cao et al. (2010) did qualitative research about the cost of refactoring and ascertained that there was a threshold beyond which the cost of modifications cannot be allowed to escalate. The development can proceed until this cost is below a threshold. Reflecting on the technical debt issue, when the agile team need to fix it, they need to think about the cost of change with a dynamic perspective, which can also be applied to balance up-front work effort and cost of

change in agile software development. As one of agile manifesto posits that people should focus more on working software over comprehensive documentation, agile practitioners are usually in confusion about how much up-front the design or architecture should be. Linking the time pressure to the confusion, agile teams intend to guess the level of upfront design in the beginning and improve it gradually. Waterman et al. (2015) give rise to six influencers of upfront design work, regarding requirement, early value, technical risk, experience, team culture, and customer agility. Although they present a dynamic system to help agile practitioner to form suitable upfront design, they neglect a key attribute, the cost of design changes. As Cao et al. (2010) ascertain, refactoring, design quality, and delivery speed interplay with each other to influence the cost of changes. Insufficient refactoring will decrease the quality of the design, and therefore accommodating change requests gets difficult over time. Even though short iterations and frequent releases reduce the magnitude of changes dramatically, the cost of change can increase quickly if the design is not cleaned up regularly (Cao et al., 2010).

To summarize, the question today that agile teams are facing is not how to stop change early in a project but how to better handle inevitable changes throughout its life cycle (J. Highsmith & Cockburn, 2001). Cost - effect balance in responding to changes is one of an insightful attribute to focus.

2.2.4 Research Trend of Responding to Changes in Agile Software Development

Recently, research about responding to changes in agile software development expanded to include more perspectives, such as emotional response, balancing tensions, and department culture. Madampe et al. (2020) posit that while agile practices may provide a framework for dealing with changes, the human aspects of responding to change need to be understood. Their pilot study shows that even though agile focuses on embracing change, agile teams undergo emotional ups and downs in responding to requirements changes. Thus, considering emotional aspects is imperative to understanding and improving how agile teams respond to changes in practice (Madampe et al., 2020). Yu & Petter (2014) applied the shared mental model's theory to a conceptual analysis of specific agile practices and demonstrate the value of agile practices

in developing shared mental models (i.e., shared understanding) among developers and customers in software development teams.

Gupta et al. (2019) categorize four different IT department cultures and combine them with two types of agile practices, contributing to the extant literature by integrating the competing values model of culture into the literature on factors affecting agile development at the IT department level. Linking agile practices with organizational context provides a new direction for academic research and practical managerial improvement. As Kaufmann et al. (2020) ascertain that emerging strategy is strongly and positively related to project success. Hence, understanding how agile practitioners perform agile activities to initiate emerging or adapting environments becomes imperative.

On the whole, the research about Agile software development has gone through a round of iteration, from the generic views, discussing agility and adaptive system, to the detailed methodologies, analyzing the effectiveness of different commercial agile methodologies, and back to the context view, arguing about specific applications in different contexts. More and more people realize that changing their ways of working to help the organization to being agile instead of doing agile is crucial to the success of agile implementation and execution. Thus, this thesis chooses a grounded theory method to understand how daily agile activities work effectively to help organizations perform better in specific project context.

3. RESEARCH DESGIN AND METHODOLOGY

Research design is the overall configuration of a piece of research involving questions about what kind of evidence is gathered and from where and how such evidence is interpreted to provide good answers to initial research question (Saunders et al., 2019). This research is guided by intrinsic interest in the practices of agile teams in responding to changes. The researcher took an exploratory approach to the topic to generate further insights and gain an in-depth understanding of the phenomena. As Bristow & Saunders (2018) explained, there is interlinkage among researcher's beliefs and assumptions, research philosophy, and research design. It is beneficial to discuss three of them to have a better understanding of the following methodology and data analysis.

3.1 RESEARCH PHILOSOPHY

According to Saunders et al. (2019), the term 'research philosophies' refers to systems of beliefs and assumptions about the development of knowledge. This means that our research philosophy contains important assumptions about the way in which we view the world. These assumptions shape all aspects of our research projects. Johnson and Clark (2006) argue that we need to be aware of the philosophical commitments we make through our choice of research strategy, since this will significantly impact on what we do and how we understand what it is we are investigating.

Based on three research assumptions, ontology, epistemology, and axiology, Saunders et al. (2019) developed five research philosophies accordingly: positivism, critical realism, interpretivism, postmodernism, and pragmatism. This thesis applied interpretivism as a research philosophy for the following reasons: firstly, the nature of reality for agile research is socially constructed with multiple meanings and interpretations. Agile values are generic guides

that can be applied and adapted to different environments. Secondly, regarding the epistemology aspect, interpretivism focuses on narratives, stories, and perceptions. New understandings and worldviews are accepted as contributions. Agile values and manifestos are new ways of understanding the same world with different interpretations. As Cohen et al. (2004) noted, agile is the focus and values behind Agile Methods that differentiate them from more traditional methods. Thus, the agile method is perceived as a new thing instead of being an innovation itself and matched with interpretivism. Finally, from an axiology perspective, agile research is a value-bound with researchers' interpretation. Different researchers with diversified backgrounds and expertise will affect the choice of research focal points. For example, researchers from the sociology field will be more inclined to interpret the agile phenomena from human interaction, cultural and organizational perspectives. The researchers from the software engineering field will be more interested in formulating the development process.

In conclusion, as posited by Saunders et al. (2019) the interpretivist's perspective is highly appropriate in the case of business and management research. Not only are business situations complex, but they are often unique, at least in terms of context.

3.2 RESEARCH APPROACH

Different approaches will be discussed to explain the choice of this thesis's research approaches. There are three main approaches to theory development: deduction, induction, and abduction. The deduction is often used to do falsification and verification. Data collection for the deduction method evaluates propositions or hypotheses related to an existing theory. Meanwhile, induction is used to build rather than test the existing theories. In inductive inferences, known premises are used to generate untested conclusions. Finally, abduction is between those two, used for incorporating, moderating, and generating theory (Saunders et al., 2019).

As noted in the literature review, there needs to be more academic research on practical agile activities in responding to changes; only some existing methods and theories can be used to test. Furthermore, in this research, insights from agile practitioners served as the foundation from which findings can be generalized on the population of interest: how to respond to changes with agile activities. Thus, reflecting on the characteristics of different research approaches, the research gap allows the researcher of this thesis to use an induction approach to structure a model in a context rather than testing the existing ones.

3.3 RESEARCH STRATEGY

This thesis applies a grounded theory approach developed by Anselm Strauss and Juliet Corbin (Corbin & Strauss, 1990) to explore the unchartered area of agile activities in responding to changes in the context of software development. Grounded theory is an inductive research strategy and approach to theory building in which knowledge is created by allowing the data to speak for itself and is well suited for exploring new concepts. The goal of grounded theory is to build a theoretical model that captures the experience of the participants in theoretical terms (Gioia et al., 2013). The model should display the dynamic relationships between the unveiled concepts describing the phenomenon being studied. Using grounded theory, we can approach the topic of interest and investigate the phenomenon from a bottom-up view without narrowing it down to quantified variables and preconceived findings (Saunders et al., 2019).

3.4 METHODOLOGY

According to Sofaer (1999), qualitative methods enable us to accurately describe the phenomena in question, using small, non-numerical, and context-specific samples to gain insights into phenomena where quantitative analyses do not provide the required depth. Qualitative research methods are valuable in providing rich descriptions of complex

phenomena; tracking unique or unexpected events; illuminating the experience and interpretation of events by actors with widely differing stakes and roles (Sofaer, 1999). While the traditional scientific method, usually quantitative, has dominated the field of research for many years and is the most widely accepted approach to theorizing, the qualitative approach is focused on concept exploration in the specific context.

The purpose of this research is to explore how agile teams respond to the changes rather than to evaluate the specific effects of a given variable on agile teams. As discussed at the beginning of the thesis, agile is an umbrella term (Schwaber & Beedle, 2002) with intensive human interactions, which can be applied in sociological and scientific fields. The influence of agile and its outcomes are challenging to quantify because human interaction has a personal and subjective quality. For these reasons, qualitative methods are deemed the most suitable way to obtain relevant insights for this research. Using grounded theory, we can investigate the elements that influence agile teams to learn to respond to changes.

Agile was adopted as the umbrella term (Schwaber & Beedle, 2002) without clear research problem or hypothesis up-front. Different organizations usually customize the agile method and encounter diversified issues while running agile, which is hard to conclude a solid theory. Meanwhile, according to Hoda et al. (2012), grounded theory (GT) is most suited to research areas that have not been explored in detail. Moreover, GT is one of the few research methods focusing on theory generation rather than extending or verifying existing theories (Hoda et al., 2012). The following sections will introduce further details of grounded theory and how to apply it.

3.4.1 Grounded Theory

Grounded Theory Origination

The Grounded theory originated from the research of sociologists Barney G. Glaser and Anselm L. Strauss, as they studied the awareness of dying patients and the role of their social status. The procedures and strategies employed in their qualitative research project led to the

formulation of the Grounded Theory method, documented in their book "The Discovery of Grounded Theory – strategies for qualitative research" (Hoda, 2022). Nearly two decades later, one of the two founding fathers, Anselm Strauss, along with Juliet Corbin, proposed the first variation of GT in their book "Basics of Qualitative Research", which came to be known as Strauss-Corbinian GT. The original GT method supported by Glaser is since referred to as Classic or Glaserian GT. Kathy Charmaz introduced a third version with her book "Constructing Grounded Theory", also known as the Constructive GT version, a decade and a half later (Hoda, 2022).

Although Classic, Strauss-Corbinian, and Constructing GT are all named with GT theory, they are formulated with different elements. Classic GT is developed based on objective epistemology and positivism research paradigm; Strauss-Corbinian GT is based on interpretive epistemology and symbolic interactionism research paradigm; Constructing GT is based on Subjective epistemology and constructivism research paradigm (Hoda, 2022). GT theory is derived from the sociological field, but more and more researchers are doing agile research by adapting GT theory for software development.

Grounded Theory Adaption in Software Development

Adaptations and updates to traditional research methods sometimes are common. For example, contextualized guidelines for other sociological research methods, such as Case Studies (Richardson & Kramer, 2006) has made it easier for software engineering researchers to apply and succeed. According to Hoda (2022), GT emerged in the sociological research scene when quantitative research focused on objectivity, accuracy, verification, and generalizability had taken a strong foothold following similar trends in the broader research communities of the natural sciences disciplines. Qualitative research, on the other hand, was seen as lacking scientific rigor and relying on anecdotal evidence. With its systematic and rigorous techniques and procedures, the introduction of GT challenged the status quo and served to re-establish the value of qualitative research in sociology. It challenged the overemphasis on theory verification

predominant at the time and argued for theory development as a fundamental research objective (Hoda, 2022).

The most successful GT in software engineering studies has been those that have overcome the entry barriers of understanding traditional GT guidelines and applied them, often with implicit adaptations, to work in Socio-technical contexts (Hoda, 2022). While no one research paradigm or worldview is correct or wrong, researchers need to be aware of and declare their philosophical stance because it influences how we conduct studies, present findings, and, most prominently, how evaluate research (Hoda, 2022). Furthermore, combining different methods together will cause issues, since there is the absence of philosophical and methodological guidelines for adaptations (Hoda, 2022). Hereafter, this thesis mainly applied Corbin & Strauss GT (Corbin & Strauss, 1999).

3.4.2 Sampling Method and Data Sample

The Grounded theory is the theoretical sampling method, which implies basing the data collection process on evolving theoretical constructs (Draucker et al., 2007). When a project begins, the researcher brings an idea of the phenomenon he or she wants to study. Based on this knowledge, groups of individuals, an organization, or a community representative of that phenomenon can be selected for study (Corbin & Strauss, 1990). To maintain consistency in data collection, the investigator should watch for indications of all critical concepts in every observation--ones carried over from previous analyses as well as ones that emerge in the situation (Corbin & Strauss, 1990).

The initial sampling was defined based on the research question of this research. The sample participants had to be 1) a person who has agile experience and 2) is currently working in one or more agile teams. By the emergence of categories, the researcher of this thesis shifted to theoretical sampling, additional sample participants were sourced iteratively until the researcher had conducted six interviews and deemed the level of theoretical saturation adequate. As the theoretical saturation point emerged, a continuation of data collection would compromise the analysis quality due to overwhelming data. According to (Chun Tie et al.,

2019), ensuring maximum variation in the sample and keeping the quantity of data for analysis to a manageable size is a great challenge. The core principle to collecting enough data is based on if the theory has been developed and proved by rationale.

Through the iterative sampling process, participants from six different projects were sourced to take part in the data collection. I explained to participants the purpose of initiating contact, briefly described the research, and gave them general information about possibilities for anonymity. According to (Robinson, 2014), individuals who volunteer to engage in "self-disclosure" must be assumed to have certain characteristics that deviate from the average member of the sample universe. In this thesis, more supportive and self-disclosed characteristics can be assumed in sampling data which may lead to self-selection bias.

The samples vary across factors to promote data heterogeneity (see Table 1). The informant roles are based on different job functionalities in an agile team, namely, a small number of people with complementary skills (Katzenbach & Smith, 1993). The team size of which participants work is from 3 to 7 people, which is typical of a scrum team. Maintaining a similar size team is due to the analysis requirement since the variables will be changed based on different sizes of team conditions, particularly communication efforts, which is a crucial aspect in agile research. Participants have diversified agile experiences because different agile maturity will contribute to differentiated views and reflections on agile values, which may help researchers to develop a comprehensive theory.

Participant	Informant Role	Team Size	Agile Experience
Participant 1 (P1)	Data Scientist	5-7	1-2
Participant 2 (P2)	Software Engineer	3-5	3-5
Participant 3 (P3)	Software Engineer	3-5	3-5
Participant 4 (P4)	UX/UI Designer	5-7	3-5
Participant 5 (P5)	Project Manager	5-7	5-10
Participant 6 (P6)	Senior Software Engineer	5-7	5-10

Table 1. *Data Sample*

3.4.3 Data Collection Method

As explained in previous sections, data collection in GT is guided by theoretical sampling, an ongoing process that helps decide what data to collect next based on the emerging theory (Hoda et al., 2011). The initial participants of this paper have relatively less agile experiences. The researcher could discern gaps by using theoretical sampling in the emerging theory, which prompted to study participants with more mature agile experiences towards later stages of research.

The Interviews Guide

This research conducted six semi-structured face-to-face (physical/virtual) interviews with open questions to agile practitioners, including UX/UI designers, engineers/developers, and project manager. Open-ended questions were used to encourage the interviewee to provide more extensive answers and reveal underlying attitudes (Grummitt, 1980). The interview questions focused on the participants' experiences working with Agile methods, particularly the challenges they faced in agile projects regarding responding to changes and how they overcame them. Accordingly, I asked practitioners the following framework questions, and the interview questions were adjusted based on ongoing interviews and were not the same every time.

#	Question
1	Can you please tell me about your project?
2	In your opinion, what are the responding to changes in the agile projects?
3	If changes occur in a project, does it have any effect?
4	How does it influence your project teams?
5	What kind of strategies are you using to handle these changes?
6	What do you think about self-organizing teams or agile teams?
7	What do you think about the relationship between responding to changes and self-organizing teams?

Table 2. Interview question lists

The interviews were voice/audio recorded according to physical or virtual meetings. Recording enables the researcher to keep detailed information. The recorded interviews were transcribed with an AI transcription tool (explained in the next section) and analyzed. Data collection and

analysis were iterative so that constant comparison of data helped guide future interviews, and the analysis of interviews fed back into the emerging results.

Conducting Interviews

The researcher introduced the purpose of the interview and essential background information to ensure that the participants could understand what direction they should talk about. Before the recording started, the researcher asked for permission from each participant and only started recording when she received positive confirmation.

The researcher conducted one pilot interview, during which she found that the questions about responding to changes needed to be more direct since the participant talked more about his own problems or issues instead of giving direct feedback regarding responding to changes. Meanwhile, the word agile value was too vague for the participant to understand. After the first pilot interview, the researcher adjusted interview questions and added the ones relevant to responding to changes to elicit the expected answers, such as what you think about requirement changes, retrospective meetings, and self-learning. The interviews were semi-structured to allow for the participants' main concerns around responding to changes to emerge.

3.4.4 Transcription Method

The interviews were recorded through audio and video recording tools to transcribe and subsequently code the collected data material. To receive better transcriptions, the researcher ensures the quality of the recordings in the given environments before conducting and recording each of the six interviews. For example, the researcher chose quiet meeting rooms instead of public resting areas. To eliminate the interviewees' concerns, the researcher gave a basic background explanation regarding how she will use the recorded data and personal information data privacy before the recording started.

The recordings were uploaded to the online transcription tool *Sonix*, which produced written transcripts marked with time stamps. When using this automated transcription tool, the written

transcript will be prone to certain errors, typically caused by factors such as the interviewees talking in an implicit accent, or the use of abbreviations. Therefore, each of the transcripts (see Appendix 1-6) were subsequently corrected to sufficient degree in accordance with the original recordings to ensure a sufficient foundation for the coding process.

While there is no academic consensus on a “best” or “most accurate” transcription style, the researcher must decide upon which style best serves the purpose based on the nature of the research problem (McMullin, 2021). Based on this perspective, the thesis employed an “intelligent verbatim” approach to transcription of quotes used in the analysis. Intelligent verbatim involves removing stutters, repetitions, and any other non-essential speech elements while preserving the original meaning of the message. By using intelligent verbatim, transcripts can be more concise and easier to analyze, while still preserving the nuances of the speaker's language and style.

Original transcript	Intelligent verbatim transcript
Um, I appreciate the value from running Agile projects in that the, the, the delivery plan becomes. I won't say it becomes more flexible, but you have a better guarantee of delivering the right product at the first go, If you execute an Agile project in the right way. Um, so that's one point, at least.	I appreciate the value of running Agile projects in that delivery plan. I won't say it becomes more flexible, but you have a better guarantee of delivering the right product on the first go, if you execute an Agile project in the right way. That’s one point.
So it's sort of a complicated statement that I'm giving you here, but it's because I believe that. There's a big difference between the delivery team not understanding the requirements sufficiently and then delivering something wrong.	I believe it is a complicated statement. There's a big difference between the delivery team not understanding the requirements sufficiently and then delivering something wrong.

It's really a matter of making sure that the, the, the sort of the continuous loop of developing agile, uh, requires you to make sure that you have a good process for when do we accept a new change or development item?	It's really a matter of making sure that continuous loop of developing agile requires you to make sure that you have a good process for when we accept a new change or development item.

Table 3. Examples Illustrating the Intelligent Verbatim Transcription Process

3.4.5 Data Analysis Strategy

As explained in this paper previously, grounded theory data analysis is consistent with open coding, selective coding and constant comparison, interacting with theoretical sampling, data collection and Memoing (see Figure 2).

Open coding is the interpretive process by which data are broken down analytically. Its purpose is to give the analyst new insights by breaking through standard ways of thinking about or interpreting phenomena reflected in the data (Corbin & Strauss, 1990). The open coding data is constantly compared with data collected from other theoretical samplings until core categories emerge.

To keep notes of the coding and some categories' emergence, *Memoing* is usually integrated into the data analysis process. According to Glaser (1978) memos are "theoretical notes about the data and the conceptual connections between categories written down as they strike the researcher". Through the memoing, and *constant comparison*, some core categories emerged. Then, S*elective coding* is applied, again, with memoing and constant comparison to delimit the coding to "only those variables that relate to the core variable in sufficiently significant ways as to produce a parsimonious theory "(Glaser, 1978, 2004).

As soon as further data collection and analysis on a particular category leads to the point of diminishing results, the category is said to have reached *Theoretical saturation* (Glaser, 1992) and can stop collecting data and coding for that category (Hoda et al., 2012). Finally, to develop the theory, we need to identify the relationships among different categories and elaborate

them based on an inductive approach since Glaser (1992) notes that theoretical coding is a property of coding and constant comparative analysis that yields the conceptual relationship between categories and their properties as they emerge.

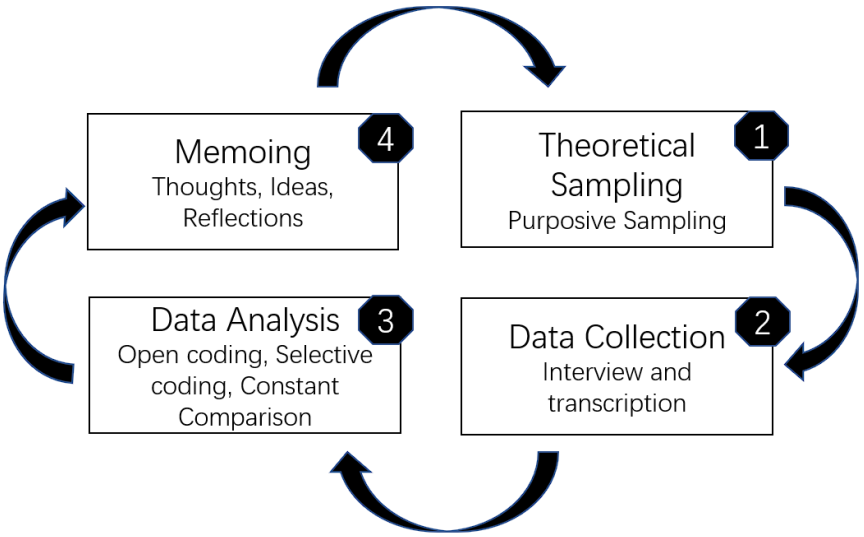

Figure 2. The procedure of data processing in GT. Adopted from Hoda (2022)

As explained in the data analysis strategy section, researchers need to follow a structured process to analyze the data in GT. In the following, the researcher explained how these processes are applied with examples from empirical data.

Open Coding

Open coding involves thoroughly analyzing the data to capture as many key points and concepts as possible (Hoda & Noble, 2017). The researcher obtained the key points from the interview transcripts. Key points are the summarized points from sections of the interview (Georgieva & Alan, 2008). Then, the researcher assigned a code to the key point. A code is a phrase that summarizes the key point in two or three words. One key point can lead to several codes. The following example can explain how the researcher coded based on raw data.

Raw data: *"So it (agile methodology) is not working in silos, which I can see it was earlier in the change management, but now it's more of a very open where everybody can see that if we follow the Azure DevOps or Agile methodology for the change, everybody is aware of where we are leading." -- P6, Senior Software Engineer*

Key point: "Not working in silos, but understanding the leading direction"

Code: *Internal Information alignment*

Code: *Azure DevOps (Agile tool)*

In the above example, Internal information alignment and Azure DevOps were codes derived from the raw transcripts, where the former summarized the idea of encouraging internal communication; and the latter capturing the idea of leveraging agile tool to enabler better information alignment.

Constant Comparison

According to Strauss & Corbin (1990), an incident should be compared against other incidents for similarities and differences in constant comparison. Namely, the codes arising out of each interview were constantly compared against the codes from the same interview and those from other interviews. Using the constant comparison method, a higher level of abstraction, concepts, can be summarized. Taking the above example, the concept that emerged was 'information alignment' which captured how agile teams respond to change to get better understanding of the project.

Concept: *information alignment*

Other concepts arising in a similar manner, here we take another example to explain it.

Raw data: "*Because when we have this sort of timely sprint, it prevents people from being perfectionists. And in this sense, improvement to me is a lot more valuable than someone trying to be or a team trying to be perfect from day one." -- P1, Data Scientist*

Key point: *"timely sprint enables continuous improvement"*

Code: *Time box, continuous improvement*

To summarize the concept, the researcher used the input from another sample raw data as below:

Raw data: *"I think it's very valuable, and it's one of the key aspects of getting value out of an Agile project is that the Agile project, if it follows certain sort of theory and models, relies on this feedback loop that you are continuously reviewing and improving your ways of working." -- P5, Manager/Scrum Master*

Key point: *"Feedback loop improve ways of working continuously"*

Code: *Feedback loop, continuous improvement*

In the first example, we understand that participants appreciate the time box of the agile ceremonies, which enables him/her to make continuous improvements. Thus, how the participant perceives the agile meetings influences his/her ability to respond to changes and make continuous improvements. Then, the researcher compared the second sample data with the first one, since both described different agile activities (sprint planning and retrospective) enable continuous improvement, and summarized the concept as below:

Concept: *enabling continuous improvement*

Then, the constant comparison method was repeated on the concept level to produce another level of abstraction called a category (Hoda et al., 2011). As the pages are limited in this thesis, the researcher did not explain how all the concepts are summarized. Instead, this thesis focused on the category level of data and explained it in detail since the final theory was based on the interrelations among the main categories.

Category

The end of open coding is marked by the emergence of a core category (Glaser, 1992). The core category is the ''main theme'' or ''main concern or problem'' for the participants (Glaser, 1978). In this thesis, the core categories emerged as Communication and collaboration acts, Adapting

acts, Balancing acts, and Learning acts. Fig. 3 shows the levels of data abstraction using GT for the detailed analysis about how categories emerged from their underlying concepts and how theory developed are discussed in the analysis part.

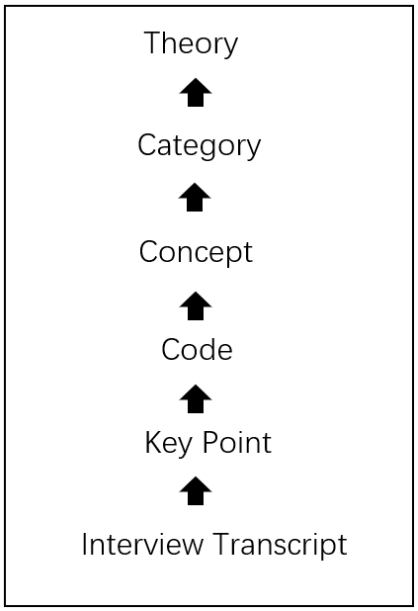

Fig. 3 Levels of Data Abstraction in GT.

3.4.6 Quality of Research

Validity and Reliability

The quality of research for this qualitative study is based on the principles of grounded theory methodology. The theoretical sampling method was used to guide the data collection process, ensuring that the evolving theoretical constructs were taken into account. The initial sampling was based on the research question, and additional sample participants were sourced iteratively until the level of theoretical saturation was achieved. The selection of participants was based on the criteria of having agile experience and working in one or more agile teams. Glaser (1997) clearly states that the focus of GT is the generation of theory and validation may be undertaken by other researchers using different methods (Hoda et al., 2012). However, Hoda et al. (2012) posit that exploring how well the theory fits with previous literature on the subject can be the source of validation of the emerging theory, which is a kind of literature

review after analysis to see what other researchers' arguments are. This thesis applied a similar way to enhance the validity of the research.

Essential concepts were identified in every observation to ensure data consistency, including ones carried over from previous analyses and ones emerging in the situation. The samples were selected from different projects and varied across factors such as informant role, team size, and agile experience. The team sizes ranged from 3 to 7 people, typical size of a scrum team. The informant roles were based on different job functionalities in an agile team, including individuals with complementary skills. The participants' diversified agile experience was also considered, as different levels of agile maturity, can contribute to differentiated views and reflections on agile values.

To address potential biases, the researcher ensured anonymity and explained the purpose of initiating contact with the participants. Additionally, the self-disclosed characteristics of the individuals who volunteered to participate in the study were assumed to deviate from the average member of the sample universe, and the possibility of self-selection bias was acknowledged. Since, as human beings, we all have inherent biases about everything. Moreover, to guard against our personal bias, the most effective strategy was to make ourselves explicitly aware of our potential biases about different situations and experiences (Glaser, 1978; Parry, 1998).

Generalizability

Software Engineering researchers applying GT often encounter criticism regarding the generalizability of their findings (Hoda et al., 2012). However, generalizability in GT is achieved not through claims of the universality of the theory but rather through the ability of the generated theory to be modifiable to fit, work, and be relevant in new and different contexts (Glaser,1992). For example, the substantive theory of this thesis about how to respond to changes in Agile software development teams can be modified to make falsifiable predictions in other substantive areas such as teams in sales and marketing. To minimize any loss or misinterpretation, all data was personally collected and analyzed by the same researcher.

Overall, it is important to note that validity, reliability and generalizability are not fixed properties of a study, but rather are ongoing concerns that must be carefully considered and addressed throughout the research process.

4. ANALYSIS

In this section, the researcher analyzed the empirical data based on the GT data analysis process and inductive research approach. The researcher selected quotations drawn from the interviews that shed particular light on the concepts. Due to space reasons, the researcher cannot describe all the underlying key points, codes, and concepts from the interviews that further ground the discussion. Meanwhile, in order to explain how different categories interplay with each other, the researcher developed a theory to illustrate the linkages among categories to answer the research question: how do small agile teams respond to changes in software development?

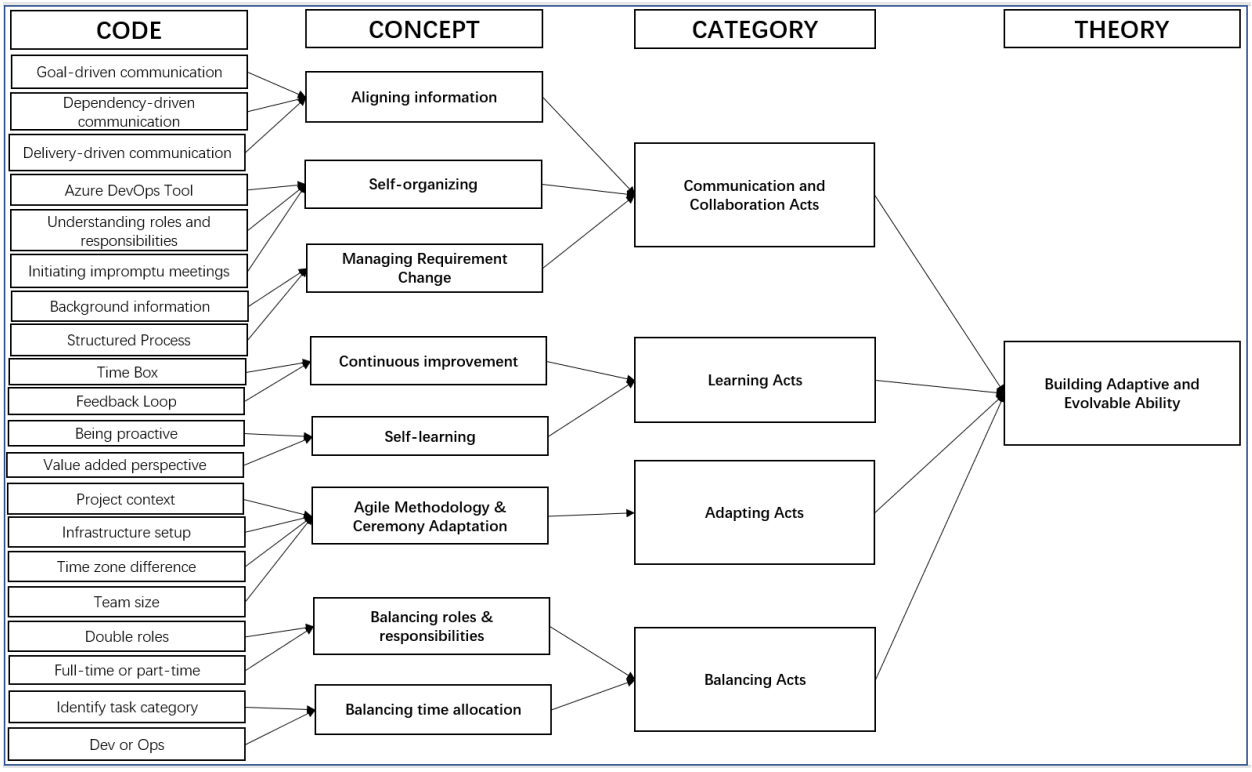

Fig. 4 Empirical data grounded from code to theory.

4.1 Communication and Collaboration Acts

Information alignment ensures that everyone on the team has a clear understanding of the project's goals, objectives, and priorities. This helps team members make decisions and take action in a way that supports the project's overall vision. Lindskog & Netz (2021) argue that when people work together, with good communication and interaction, they can work at significantly higher levels of efficiency than when they use their individual capacities.

"It (improvement work items) can be updating the scripts, updating the comments or writing SOPs or creating test plans. So, you will only get that idea once you know where the project is heading, what the end goal we are trying to achieve, and what is at stake." -- P6, Senior Software Engineer

Secondly, ***Self-organizing*** enables the team to work together to solve problems and make decisions collaboratively. By establishing clear communication channels, shared goals, and objectives, self-organizing teams can effectively work together toward common goals.

"And I think each of us understands our role and how we can get help from others. If you need more information, we can say could you please make a plan, for example, mapped out some processes? XXX, we don't understand this, can we have a meeting to understand?" -- P3, Software Engineer

Requirement change management ensures that the team can adapt to changing project requirements. This is essential in an Agile environment where requirements may change frequently. The team needs to be able to respond quickly and effectively to the changes while still delivering value to the customer. Requirement change was used by agile participants to initiate more frequent communication.

"And if the change is well described and the process of managing the change or delivering the new feature or updating the feature is in control, then it's smooth sailing, and you just on a regular basis in your agile iterations or what you call them sprints or something like that. Then you can just take on new requirements or changes on a regular basis." -- P5, Project Manager

Based on the theories of aligning information, self-organizing, and managing requirement change, the category emerged as **Communication & Collaboration acts**. The reason to combine communication and collaboration together is due to the argument from (J. A. Highsmith, 2000), communication is somehow passive, which transfers information. And collaboration is active, which requires active participation, with the intent to add value. The combination of both can describe agile practices more comprehensively.

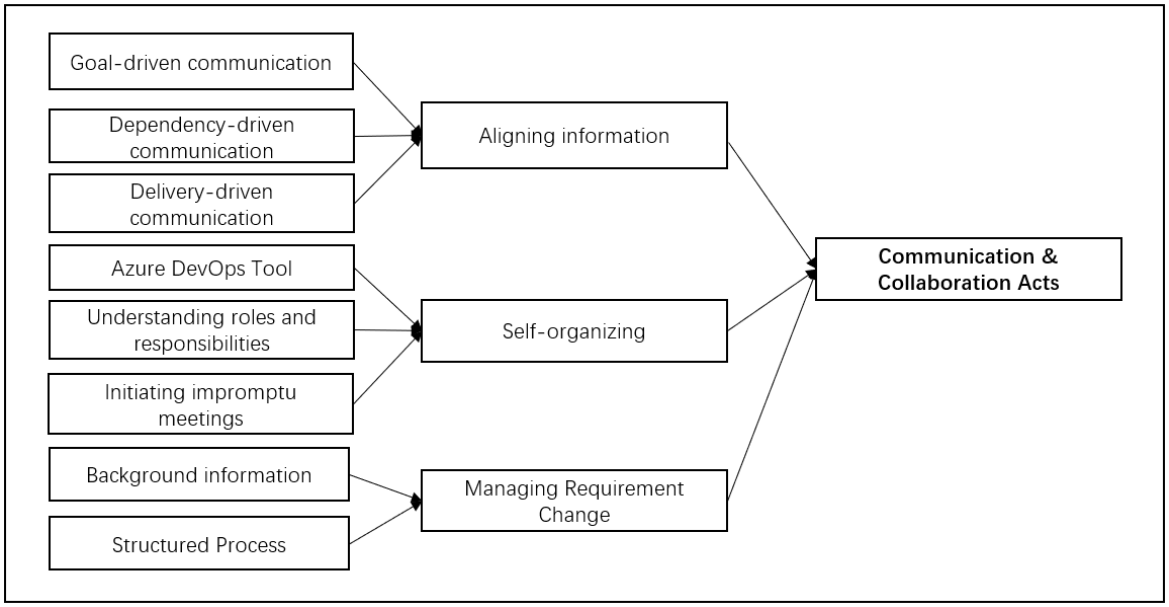

Fig. 5 An example of category “Communication and Collaboration Acts” emergence from code.

4.2 Learning Acts

The second category of this thesis is Learning Acts, which includes the concepts of continuous improvement and self-learning, emerging from the specific codes, feedback loop, time box, being proactive, and value-added perspective (Fig. 6). Feedback loop and time box relate to the regular review and retrospective agile activities, through which, changes, issues, and, challenges are identified and discussed, enabling continuous improvement.

Moreover, when the team members have a clear view of what they are doing or going to do (linking back to the communication and collaboration category), they can understand what

value they are able to provide so that become more proactive. The whole process is a positive cycle that enables the agile team to learn and develop continuously.

"And they weren't proactive before starting the Agile method working, in Sprint. But now I can feel like they understand better what we are doing, and they propose more solutions that are for me proactive because they propose solutions, meaning that they are thinking about the solution and working towards a common goal." – P3, Software Engineer

"The way I thought is that okay when I'm done with my deliverables, there is something which can be done in the later part. So instead of that, let's focus on something which can add value. Things will not come when somebody doesn't have that understanding. If somebody is like, I need to do ABC, done? No, but there are more. You can do A.1, can do B.2. So, you can also do the subparts. Once somebody has a clear picture, they can think more and come up with this improvement item." -- P6, Senior Software Engineer

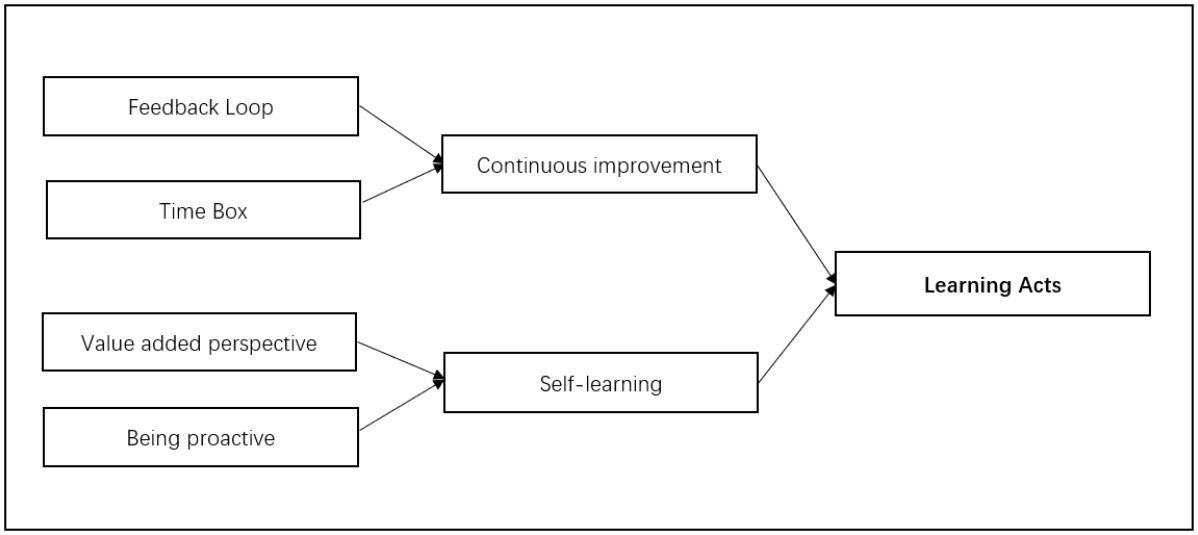

Fig. 6 An example of category "Learning Acts" emergence from code.

Furthermore, continuous improvement and self-learning are complementary concepts that support each other (Fig. 7). ***Continuous improvement*** encourages individuals to do self-learning and seek opportunities to develop new skills and knowledge. ***Self-learning*** supports individuals in gaining new insights and knowledge that can be applied to continuous improvement efforts,

especially for tacit knowledge, which is grounded in knowing-in-action and knowing-in-practice (Hoda et al., 2013). Self-learning can also help individuals identify areas for improvement and develop new solutions to existing problems.

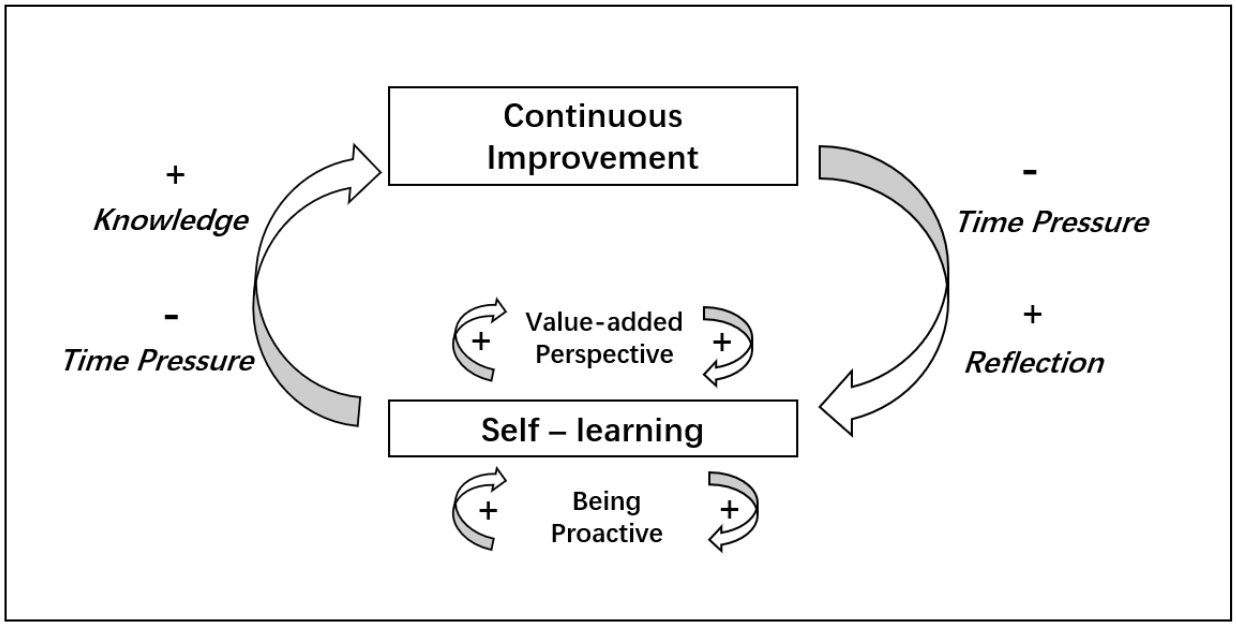

Fig. 7 Interaction between continuous improvement and self-learning.

However, during the interview, the researcher found a negative variable in continuous improvement and self-learning circle: the time pressure of iteration. When the agile team members feel the time pressure, they intend to skip or postpone the self-learning, reducing the knowledge output, causing learning debt as posited by Hoda et al. (2013).

"…things are going a little bit fast because you have two weeks in the beginning. I invest my time in understanding the tasks that I have, and send the invites and prepare for the invites without even realizing it (iteration) is the end. And then you're over and over, and I don't really dedicate the time for improvements in the way of self-reflecting and self-learning together with the team as well." -- P4, UX Designer

4.3 Balancing Acts

Balancing roles and responsibility are crucial for ensuring the team is equipped to respond to changes in an agile project. By ensuring that each team member understands their responsibilities and can work collaboratively to respond to changes, the team can adapt quickly and effectively to change project requirements as well as the changes in roles and responsibilities. The study by Conboy et al. (2011) found that, compared to plan-driven software development, the boundaries between the developer roles were less defined in the Agile way of working, demonstrating the criticality of understanding the roles and responsibility.

"I think each of us understands our role and how we can get help from others. For example, I know that as a lead engineer, I need to interact a lot with XXX..." -- P3, Software Engineer

As the project progresses, some changes emerging, team members need to take on new responsibilities or adjust their roles to meet project requirement. For instance, in the initiating phase, the project team was only a 2-3 people team, so there was not necessary to have a full-time scrum master. The team leader took that responsibility. However, as the project proceeded, more people joined and needed more communication and facilitation work. A full-time scrum master was needed. Melo et al. (2011) pointed out that small and mixed teams, with the required expertise, and full-time allocated to the project, are the most desirable attributes of team composition. However, not every project is executed in an experimental environment; namely, practitioners cannot control all variables. What they can do is to observe dynamic situations continuously and balance variables accordingly.

"But I was worried because I feel like I don't want to spend much time on those kinds of tasks as a scrum master because then I lose focus on my other tasks. Yeah. So, I prefer someone specialized in it." -- P3, Software Engineer

Adjusting time allocation plays an important role in Balancing activities to maximize team's output dynamically. For instance, when the team is in the development phase, bug-fixing time is less than in the operation phase.

"If bugs come in or the application goes down for one way or another, there's nothing sort of shielding us from that. So, while we are doing new features, then we can be pulled out because now this bug, then we need to look at the bug immediately because we are both in development but we also in operations." -- P2, Software Engineer

Moreover, work item category influences time allocation when agile practitioners take actions. For newly developed feature, agile practitioners usually spend more time on developing and delivering. For improvement work items, agile practitioners prefer to spend more time on learning to maximize the output.

"So, in that sense, the amount of time that goes into learning versus actual development, I would say maybe 30% is learning 70% of that time in the enabler is delivering or developing. Okay. But when we go to the improvement stage, where we come back to an old enabler and then we try and work on something, the learning there would take a bigger percentage … the learning part in that phase would take a swap. So, it will be 70% learning now and then 30% actually improving on what I have done or writing a new set of codes, for example." – P1, Software Engineer

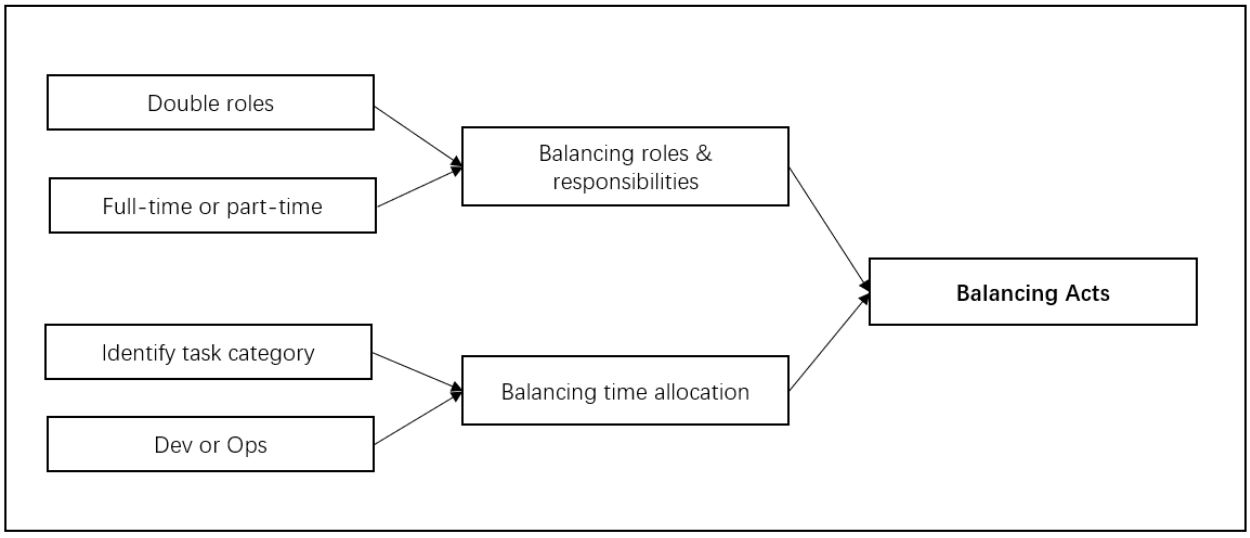

Fig. 8 An example of category "Balancing Acts" emergence from code.

4.4 Adapting Acts

Agile methodology and ceremony adaptation work together to provide a flexible and adaptable approach to managing change in an Agile project (See Fig. 9). When teams adapt the Agile methodology and ceremonies to fit their needs, they can more easily collaborate with stakeholders and communicate about project progress. Agile methodology adaptation is usually based on the project context. For example, the project size influences the methodology adoption, the small project is more suitable to adopt Scrum, and a large project is more suitable to use the SAFe.

"We are trying to build all the (Scrum) blocks progressively. Not all at once because we don't have a scrum master. So, we are trying to adapt it to our way of working. But I think it improved for sure the collaboration and the understanding of the project." -- P3, Software Engineer

"We have light agile in the sense that it doesn't make sense to roll out the full thing because coordinating between four people and the total is a very easy job. so we do have dailies. We are struggling a bit with finding the right balance between sort of having all the official meetings, like what they call iteration planning, and all those reviews and all." – P2, Software Engineer

Agile ceremony adaptation helps teams to communicate according to their priority or physical condition. For example, after an important milestone, a longer retrospective is required. Or due to the time zone difference, all the meetings need to be set up at a specific time by virtual meeting software.

"But it's difficult for Indian team, because we have a difference in time zone. We cannot meet physically, so we need to improve this communication, and I think the agile planning and the sprints have helped a lot. Also, the structure for having standups, even though it's 15 minutes, gives a clear idea of if they are struggling or not. And they take those 15 minutes to ask for help." -- P3, Software Engineer

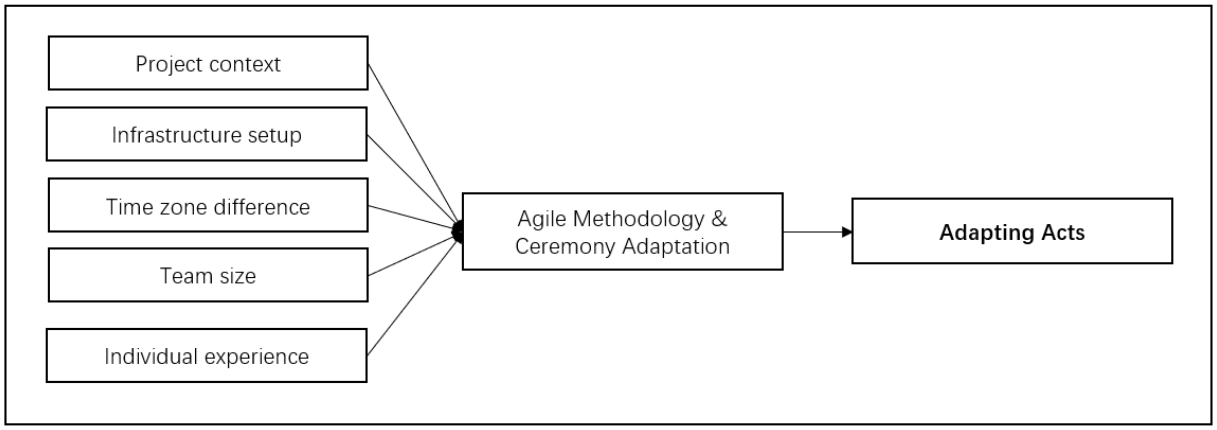

Fig. 9 An example of category “Adapting Acts” emergence from code.

Glaser (1978) posited that the core category guides further data collection, analysis, and theoretical sampling. The researcher started to do selective coding by adjusting the interview questions to be more specific to the core category. For example, the researcher modified the interview question *“Which kind of activities ... do you think the most important, efficient or effective, or valuable? (P2)”* into *“Regarding the internal team members, if you figure out the changes that happened during the iteration, how will you communicate with each other and figure out the solution? (P6)”.* When further data collection and analysis on a particular category leads to the point of diminishing results, the category is said to have reached Theoretical Saturation (Glaser, 1992). After that, the researcher stopped collecting data and coding for that category and moved forward, analyzing the relationship among the main categories.

4.5 Theory Development - Theoretical coding

The final step of GT’s data analysis process is theoretical coding which involves conceptualizing how the categories relate to each other and formulating a set of interrelated hypotheses to be represented as a theory (Glaser, 2005). Glaser (1992) lists several common structures of theories that are known as theoretical coding families. Some of these include The Six C’s,

Process, Degree family, and Dimension family. Researchers need to choose the most suitable one to do GT analysis. In this thesis, the researcher applied dimensional analysis to analyze the relationships among four categories since I need to identify the underlying dimensions influencing or impacting the phenomenon of interest. A grounded theory is more than just a set of descriptive categories: it should also describe the key relationships between those categories, i.e., a set of interrelated hypotheses (Glaser, 2005).

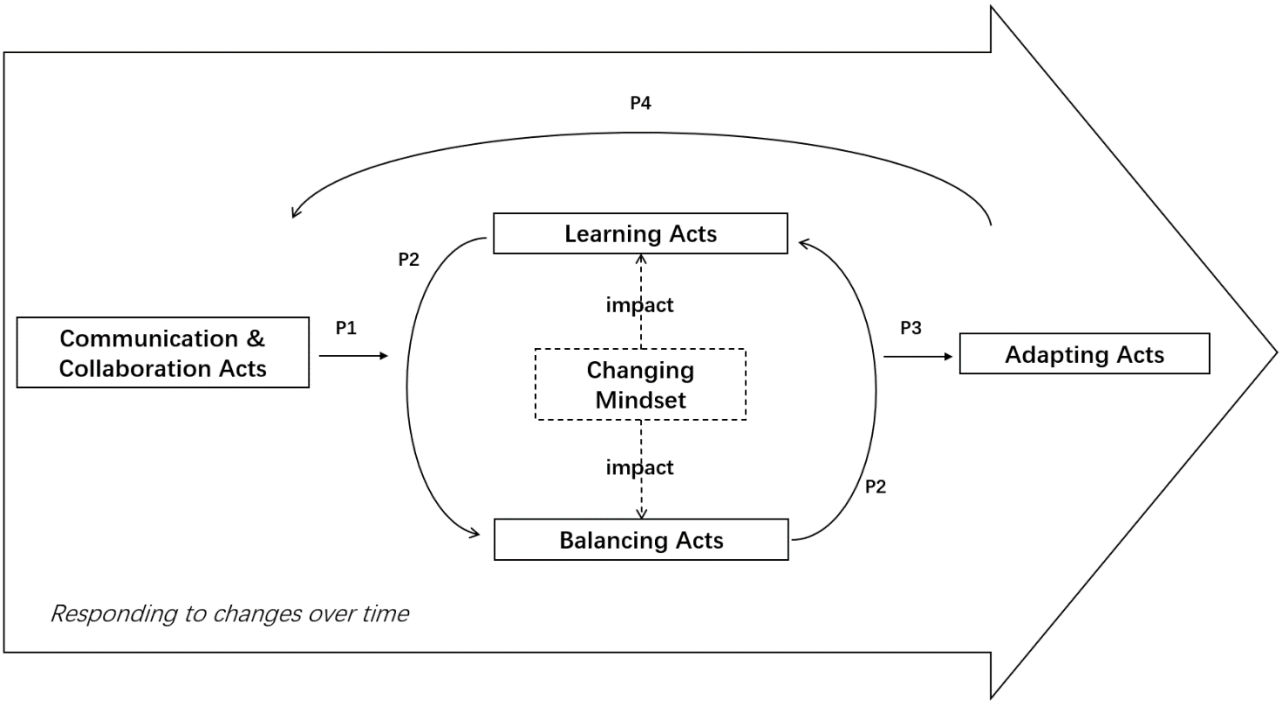

Fig. 10 A Grounded Theory of agile team's acts as an adaptive and evolvable system while responding to changes.

Figure 10 shows the dimensions in theory (within solid boxes) as described in the previous section and their inter-relationships between dimensions (propositions P1-P4, represented by arrows). We now discuss these propositions in detail:

P1: T*he team's communication and collaboration abilities are necessary for the changes in which the team's learning and balancing acts occur.*

The changes in the team always start from the communication and collaboration process, which is performed as an input data source. Whether it regards the goal change or requirement

change, the agile team needs to learn how to respond to the changes based on the input data from communication and collaboration. Good communication and collaboration enable agile teams to identify problems or seize opportunities to boost their improvement. For instance, as explained by P2 software engineer, they gave rise to a solution to manage ad hoc bug-fixing tasks by allocating specific hours per iteration to fix bugs. This bug-fixing time is the output of the retrospective meeting (communication and collaboration). Meanwhile, the agile team balanced what they learned from learning acts since the agile framework is dynamic and comprehensive. The interrelationships among different parts need to be balanced with a systematic view. On this occasion, the agile teams communicated and collaborated together to know how much capacity they had, and what priorities were with the pending development features to develop specific time allocation (balancing acts).

"We had many ideas. But what really worked was that we set time aside, and then we looked at each other like we were trying to see how much time we had spent on bugs previously, to see what sort of baseline for the bugs was. So, we registered all bugs and sort of put the time on them that it took. But it was only later that we came up with the idea to reserve time for it." -- P2, Software Engineer

P2: *The learning and the balancing acts tend to reflect and adapt to each other to enable the responding to changes.*

Learning acts enable the agile team to observe, identify, reflect, and implement. Individuals in the agile team tend to figure out many solutions, new ideas, or technologies and implement them all at once. Nevertheless, the reality is that the team's capacity and ability are limited. The agile team needs to introduce balancing acts to tune the allocation so all stakeholders can get maximum output. The balancing acts are not easy to perform since, in a multi-dimensional, comprehensive, and dynamic system, the agile team should monitor variables as frequently as possible in the project environment to have a whole picture and clear priority to facilitate necessary communication and collaboration. Among these balancing acts, the negative outputs

of tuning activities (unbalanced events) will be the input of learning acts, to be studied and improved as long as the agile teams have the sense of making continuous improvement.

"We put ten hours aside, and if something comes in, then you know this bug, it took me two hours to fix. Then, I took two hours out of that pool, and that's eight hours left for the rest of the sprint for people to do something." -- P2, Software Engineer

P3: *The changes in the learning acts and the balancing acts are necessary for adapting acts.*

When the agile team figured out the learning acts and formulated the balance of relevant aspects, the team will be able to adapt acts, adjusting the suitable agile methodology or ceremonies. Software developers rarely adopt systems and software development methods outright, but rather filter and combine elements from these methods to fit their needs (Babb et al., 2014). Learning acts and balancing acts interact with each other and generate new input content for adapting acts. For example, the length of iterations and the frequency of daily standup meetings could all be the input for adapting acts to help agile teams build an effective framework. Adapting acts without learning and balancing support will be a castle in the air. If the agile practitioners sit together to think and discuss adapting solutions without input from teams' self-reflections, no highly effective results will be obtained.

"For one project we discussed extending, making the iteration bigger because we felt that the Agile meetings made sense, but they took up too much of our time. So, instead we wanted to extend by one week or so we would have more time in between the meetings." -- P2, Software Engineer

P4: *Adapting acts enable further communication and collaboration acts to make changes occur.*

Adaptation is a pattern of continuous change that exists in contrast to the periodic discrete change process found in many organizations (J. A. Highsmith, 2000). While implementing the adapting acts, the agile team keeps observing the effect of newly added adapting acts and

generates more adapting proposals for responding to new changes through initiating a round of communications and collaborations. This practice helps the agile team generate a positive circle to move forward and always be able to respond to changes.

"It (agile framework) can be adapted to your team because we are not following the structure by the book. We're just taking what it can work for us because we are a smaller team. And by taking those and adapting it to ourselves. Of course, right now the structure is not the final one. So, we are still adapting it." -- P3, Software Engineer

Meanwhile, values and principles shape the ecosystem. Without a set of stated values and principles, an ecosystem is sterile, reflecting practices but not the people interacting within it (J. A. Highsmith, 2002). Highsmith (2000) argues that how we think of this world, stable or unpredictable, shapes our responses, control, or collaboration. Butler and Gray (2006) posit that mindfulness involves openness to novelty, alertness to distinction, sensitivity to different contexts, awareness of multiple perspectives, and an orientation in the present, which are required by different agile acts in an adaptive agile system. The adaptive system of responding to changes in this thesis is driven by an essential spirit, changing mindset, since agile team members need to accept and embrace changes while learning and balancing their work to create emergence.

"I think I've always embraced agile ways of delivering. Even before I started working on Agile projects. And the reason for it is that it allows for change." -- P5, Project Manager

In conclusion, the theory depicted in Figure 10 is not a linear progression of events for agile teams to respond to changes. Instead, the theory describes a complex network of multi-dimensional acts that occur over time as a team responds to changes and needs to be discussed with a systematic view.

5. DISCUSSION

This thesis sets out to bridge a gap in agile literature by investigating responses to changes in agile teams by collecting daily activities, holding a systematic perspective. The research has thus been guided by the following research question: How do agile teams respond to changes? By conducting a grounded theory study based on interviews with six agile practitioners in the software development field, the researcher has established the system that agile teams learn and balance agile activities based on communication and collaboration to make adaptations through a process consisting of multiple components. The researcher found that agile teams learn through communication and collaboration, which are affected by various conditional factors, such as subject-driven communication, the level of self-learning, and the structured way to manage requirement change.

The learning and balancing complementary mechanism generate outcomes in the form of tacit and explicit new knowledge, which guides adaptation acts. The interrelations of different acts and outcomes together form the basis for a dynamic and adaptive system of responding to changes in agile project, as presented in the analysis. In the following sections, the researcher described the related work, the implications of the results to theory and practice, the limitations of the study, and the potential research for the future.

5.1 Related Work

As posited by Glaser (1978), the purpose of a major literature review after analysis is to (a) protect the findings from preconceived notions and (b) relate the research findings to the literature through the integration of ideas. In this section, the researcher discussed the acts for responding to changes and how they interact with each other. Firstly, the researcher discussed how the communication and collaboration of agile teams enable learning and balancing acts to

respond to changes. Then the researcher discussed how agile teams' learning and balancing acts interact with each other to make adaptations for responding to the changes. Finally, the researcher discussed how adaptation enables communication and collaboration. The main works of literature used as input for discussion are *Coevolving Systems and the Organization of Agile Software Development* (Vidgen & Wang, 2009) and *Adaptive Software Development: A Collaborative Approach to Manage Complex Systems* (J. A. Highsmith, 2000).

Communication and collaboration with learning and balancing acts

Communication and collaboration in this thesis are grounded in the categories of information alignment, self-organizing activities, and requirement change management. Communication and collaboration among internal team members and between internal members and external customers are different. For internal members, following teams' agile disciplines will help them to be autonomous and self-organized since under the loose structure (general agile guideline, not imposed by the management team), team members decide their own working styles and rhythm based on the team's disciplines, such as daily standup or peer review, and impromptu meetings, to create a favorable environment to interact (Vidgen & Wang, 2009). Those interactions are the information flows and balancing them will create an environment conducive to emergence (responding to changes) (J. A. Highsmith, 2000).

Communication and collaboration between agile teams and customers are slightly different in the aspects of information alignment and change management. Collaboration within a single feature team is enhanced by improving interpersonal skills. Collaboration across multiple teams is enhanced by creating an adaptive culture environment and building effective structural support systems (J. A. Highsmith, 2000). Vidgen and Wang (2019) posit that a close relationship between a team and their customers is needed such that developers understand the (changing) business environment and customers understand the (changing) capability of technology, and each can apply an informed selection pressure on the other. On this occasion, Vidgen & Wang (2009) argue that up-front specification of requirements will inhibit the agile system from responding to changes. However, reflecting on the data I collected in this thesis, the up-front

specification of requirements can be used as the input of communication and collaboration rather than being followed by the book, depending on how the agile teams balance them.

As illustrated in Fig. 10 (agile system), the researcher argues that communication and collaboration are prerequisites of balancing and learning acts. Vigen & Wang (2009)'s statement supports my argument that they posit that learning emerges from team members' interactions (communication and collaboration). The agile loose structure provides an enabling context to sustain self-organized team activities and facilitate sharing and learning among team members (Vidgen & Wang, 2009).

Through effective communication and collaboration, team learning is developed as a collective result whereby a team acquires new knowledge and competencies as a result of individual learning being shared among team members. And teams share knowledge about the project as well as their understanding of the working context. The more knowledge the team learns from the project and the context, the more emergences of responding to changes will appear. Eisenhardt & Galunic (2000) also emphasize the importance of creating a specific working context that enables business units (agile teams) to emerge collaborations rather than define a structure to control them in a traditional way. Learning acts can be categorized into exploitation and exploration ways.

Balancing acts are influential elements for the team to respond to changes. Vidgen & Wang (2009) state that a truly agile process is a delicate balance of stability and uncertainty, enabling a software development team to work adaptively at a fast yet sustainable pace. Stability is a baseline or mechanism that the team can follow to avoid too much information going through them. Uncertainty is a pool of resources that can be used as input for learning and balancing acts. Agile teams need to balance stability and uncertainty with changing minds to maximize output. The balancing acts happened in many fields in agile context. For instance, how to balance iteration length. Reflecting on my analysis in the previous section, suitable iteration length is developed by the learning and balancing acts. Vidgen & Wang (2009) 's argument that a suitable pace strikes a balance such that the iteration cycle is long enough to get some meaningful work done but short enough not to lose momentum and responsiveness to change

also supports my point. Eisenhardt & Galunic (2000), have a similar argument that they (coevolving companies) balance the tension between too many links that restrict adaptation and too few that miss important opportunities for synergies. Balancing rigorous and flexible practices is also an important activity. Some parts of the development effort are more certain such as configuration control. Other parts of development are more uncertain and unstable such as speculation. The ability to judge what tool to use in which situation is a necessary skill for the team leader (J. A. Highsmith, 2000).

Learning and balancing acts drive the adaptations

Agile adaptation has been made by identifying roles, practices, artifacts, and processes that need to be suitable for current situations. The situation is identified with different factors related to team, internal and external environment, objectives, maturity levels, and previous knowledge (Kalus & Kuhrmann, 2013). In a strategic view, the agile adaptations are the ones that guide or facilitate agile teams to respond to changes and uncertainties, such as creating environments in which ideas flourish, risks are taken, and mistakes are viewed as learning opportunities (J. A. Highsmith, 2000). From an executive perspective, these adaptations are about how to adjust roles and responsibilities, ceremonies, processes and others. The identifying process is about learning and balancing, taking roles identifying as an example: 1) learning what roles take which responsibilities helps agile teams make more effective communications since team members have clear communication channels. 2) Balancing roles on individual and among different team members enables the agile team to evolve continuously since the project context and environment is uncertain and unstable. However, how to adapt agile methodologies is a great challenge.

Cao et al. (2009) adapted AST (Adaptive structuration theory) framework to address the issue. They argue that agile methods provide a structure for software projects, and an adapted structure emerges due to the social action of the participants in the process (Cao et al., 2009). Social action involves using technology in ways that influence and are influenced by social structures and agents' actions. Cao et al. (2009) emphasize the importance of the project and

organizational context and mutual interactions among stakeholders when adopting agile methodologies.

Furthermore, Vidgen & Wang (2009) focus on the process perspective and argue that the process needs continuous adjustment and adaptation to avoid rigidity and deterioration. The agile team needs to adapt the process to the development context taking into account factors such as the type of application and the customer (social action). Reflecting on the data of this thesis, one of the agile methodology adaptations refers to the length of iteration, which is suitable for one customer but will be inappropriate for another customer. Meanwhile, the adaptation process is not static; regularly reviewing is necessary since the reviewing of process allows a team to take gradual steps to change and improve rather than leaving it to a stage where no effective action can be taken. The reviewing process is a learning and balancing process, during which agile teams make decisions to choose the right adapting solutions. Reviewing is not merely a passive response to change but an active one seeking opportunities for change (Vidgen & Wang, 2009).

Aditionally, agile methodology is not an absolute revolutionary in software development, aiming to turn over everything. When the agile teams try to figure out solutions to adaptations, they also need to identify which process should be adapted and which should not be. As Highsmith (2000) posits, an adaptive (agile) project manager knows when to use rigorous processes and when to use more flexible or problem-solving ones. Highsmith (2000) gives an example by asking the agile teams what the most predictable problems in the project and receiving the answer is configuration problems. Then, putting a more rigorous process in place for configuration problems will be all this project needs to put it back into balance (J. A. Highsmith, 2000). However, developing sufficient judgment to balance rigor and flexibility may be the most difficult adaptive skill to learn... continuous adaptation is the only strategy that works (J. A. Highsmith, 2000).

Adaptations enable communications and collaboration

Adaptation is not only about adjusting the agile methodologies but also creating adaptative culture and environment so that the agile teams can get the most out of the agile values. Adaptation is significantly more critical than optimization since it enables the emergence, namely, the arrival of the fittest, and self-organization (J. A. Highsmith, 2000), which is the core category of communication and collaboration. Any new emergence from adaptation, especially from the implementation process, may initiate new communication and collaboration.

Furthermore, Highsmith (2000) posits that adaptation helps tune collaboration networks since the degree of adaptation decreases above certain connections. For example, according to the interview data of this thesis, the agile team assigned a scrum master role to the team leader to facilitate collaboration, and the agile team ran well in the beginning. However, with the development of the team, more team members joined, the previous adaptation needed to be more suitable and tune the collaboration way by adding one role taking responsibility for facilitating collaboration.

On the whole, the related literatures support the theory of this thesis, ascertaining communication & collaboration, learning, balancing and adapting acts are interconnected and vital elements in agile software development. By fostering effective communication and collaboration, teams can enhance their ability to learn, adapt to changes, and deliver high-quality software products in a dynamic and evolving environment. The continuous process of adaptation enables teams to optimize their practices, roles, and processes, ultimately leading to improved project outcomes and customer satisfaction.

5.2 Managerial Implications

In addition to the theoretical implications previously presented and discussed in this thesis, the research also entails certain managerial implications for agile teams in the software

development field. The findings provide insights into how agile teams respond to changes in software development. The study based on empirical data analysis and related work discussion can provide guidance and an expanded understanding for agile team members.

The thesis developed an adaptive agile system based on grounded theory and presented four categories of agile activity that lead the small agile team to respond to changes through constant interactions. As an individual in the agile team, 1) understanding the core dimensions that enable changes helps team members inspect their daily work; 2) perceiving agile activities in a systematic and dynamic way enables agile teams to make rational decisions in the changing environment. For example, there are many different agile activities based on different agile methodologies, and if team members implement all of them without balancing sense, failure will be an inevitable end. 3) understanding the importance of changing mindset in responding to changes helps the agile team take more effective and efficient actions since emergent orders always derive from an adaptive mindset (Highsmith, 2002).

Besides, the management team can benefit from understanding how individual team respond to changes to build equilibrium agile organization environment to explore and exploit new opportunities (Kaufmann et al., 2020).

5.3 Limitations

The inherent limitation of a Grounded Theory study is that the resulting theory can only be said to explain the specific contexts explored in the study. Since the codes, concepts, and categories emerged directly from the data, which in turn was collected directly from the real world, the results are grounded in the context of the data (Adolph et al., 2008).

The results presented in this paper are from participants who are all currently performing Agile practices within self-organizing agile teams, and the team size is small. Although we have one participant from the management team, most are individual agile team members. Meanwhile, as all the participants are from individual development teams (engineers, UX, scrum master),

there is not much relevant content from a business execution perspective. Thus, our grounded theory is developed in a specific context, at the development team level instead of the whole enterprise organizational level. This limitation is mitigated by explicitly explaining the project context and choosing the appropriate research method (GT), which is specialized for qualitative research within specific contexts.

Generally, the researcher of this paper does not claim this theory to be absolute or final. The extensions to the theory based on uncovered aspects or finer details of the present dimensions or potential discovery of new dimensions from future studies are more than welcome.

5.4 Future work

As illustrated in theory, changing mindset influences learning and adapting acts. It lays a foundation for an adaptive system in responding to changes, but how to practically build a changing or adaptive mindset needs to be discussed more. Thus, understanding how to cultivate changing mindset in daily life will be a direction for future research. Furthermore, as the researcher discussed in the previous section, communication and collaboration are key categories for agile teams to focus on while responding to changes. Currently, many new ways of communication and collaboration are emerging, such as Chatgpt, OpenAI, which will definitely change the ways of communication and collaboration. How agile teams respond to changes with new emerging technologies will be a further topic to discuss. Lastly, although this thesis ascertains the theory that can be used to explain and guide agile teams to respond to changes, the effectiveness of the theory has yet to be tested. Other researchers can apply further research to test and validate the theory.

6. CONCLUSION

This thesis explores a gap in the agile software development literature by conducting an exploratory, grounded theory study investigating the complex issue of how agile teams respond to changes. By conducting a qualitative study based on interviews with six agile practitioners, the researcher was able to provide an answer to the following research question:

How do small agile teams respond to changes in software development?

The main findings constitute the development of a grounded theory model establishing the adaptive and evolvable system in which agile practitioners develop the ability to respond to changes. The adaptive and evolvable system consists of four main domains: communication and collaboration, learning, balancing, and adapting. Communication and collaboration among agile team members provide input for learning and balancing acts. Learning and balancing acts are complementary performing as a circle to influence each other. Agile teams keep changing their mindset, enabling them to develop positive learning and balancing circles. Agile teams come to a conclusion about adapting based on the learning and balancing circle. Further communication and collaboration will emerge during or after adapting acts were taken. Continuously, agile teams develop their abilities to respond to changes in the adaptive and evolvable system.

Subsequent to the analysis, the thesis offers a discussion of the findings, using the relevant literature to support the propositions. To some extent, the links among different propositions are reflected in some of the most frequently cited papers. Thus, the grounded theory of this thesis contributes some insights regarding adaptive and evolvable systems in agile software development. Further, this thesis discussed the potential for developing an agile team's ability to respond to changes by analyzing team members' emotions and newly emerged communication and collaboration methods.

7. REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile Software Development Methods: Review* and *Analysis*. http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf.

Ågerfalk,P., Fitzgerald, B., Slaughter, S. (2009).Introduction to the special issue: flexible and distributed information systems development: state of the art and research challenges. Information Systems Research 20,317.

Andreessen, M. (2011). Why software is eating the world. Wall Street Journal. (August 20). Accessed August 19, 2015, http://www.wsj.com/ articles/SB10001424053111903480904576512250915629460.

Adolph, S. (2006). "What lessons can the agile community learn from a maverick fighter pilot?" in Agile Conference (AGILE '06).

Adolph, S., Hall, W., Kruchten, P. (2008). A methodological leg to stand on: lessons learned using grounded theory to study software development, in: CASCON'08, ACM, New York, 2008, pp. 166–178.

Babb, J. S., Hoda, R., & Nørbjerg, J. (2014). LNBIP 186 - XP in a Small Software Development Business: Adapting to Local Constraints. In LNBIP (Vol. 186).

Beck, K. (1999) "Embrace change with extreme programming", IEEE Computer (Oct. 1999)70–77.

Beck, K., Cockburn, A., Jeffries, R., Highsmith, J. (2001) "Agile manifesto", http://www.agilemanifesto.org, 2001, 12-4-2002.

Boehm, B., "A spiral model of software development and enhancement", IEEE Com-puter21(5) (1988) 61–72.

Butler, B. S., P. H. Gray. (2006). Reliability, mindfulness, and information systems. MIS Quart.30(2) 211–224.

Boehm, B. W. (1976). Software Engineering. IEEE Transactions on Computers, C–25(12), 1226–1241. https://doi.org/10.1109/TC.1976.1674590

Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. European Journal of Information Systems, 18(4), 332–343. https://doi.org/10.1057/ejis.2009.26

Cao, L., Ramesh, B., & Abdel-Hamid, T. (2010). Modeling dynamics in agile software development. ACM Transactions on Management Information Systems. https://doi.org/10.1145/1877725.1877730

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. Information Systems Research, 20(3), 329–354. https://doi.org/10.1287/isre.1090.0236

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. Information Systems Research, 20(3), 329–354. https://doi.org/10.1287/isre.1090.0236

Cockburn, A., Highsmith, J. (2001). "Agile software development: The business of innovation", IEEE Computer (Sept. 2001) 120–122.

Cockburn, A., (2002). “Agile software development joins the ‘would-be’ crowd”, Cutter IT Journal (Jan. 2002) 6–12.

Charmaz, K., (2006). Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis. Thousand Oaks, CA, USA: Sage.

Chun Tie, Y., Birks, M., & Francis, K. (2019). Grounded theory research: A design framework for novice researchers. SAGE Open Medicine, 7. https://doi.org/10.1177/2050312118822927

Corbin, J., & Strauss, A. (1990). Grounded Theory Research: Procedures, Canons, and Evaluative Criteria. In Qualitative Sociology (Vol. 13).

Davies, I., Green, P., Rosemann, M., Indulska, M., & Gallo, S. (2006). How do practitioners use conceptual modeling in practice? Data and Knowledge Engineering, 58(3), 358–380. https://doi.org/10.1016/j.datak.2005.07.007

Denning S. (2013). Why Agile can be a game changer for managing continuous innovation in many industries? Strategy & Leadership. https://doi.org/10.1108/10878571311318187

Denning S. (2016a). How to make the whole organization “Agile”? Strategy and Leadership. 44(4).

Draucker, C. B., Martsolf, D. S., Ross, R., & Rusk, T. B. (2007). Theoretical sampling and category development in grounded theory. Qualitative Health Research, 17(8), 1137–1148. https://doi.org/10.1177/1049732307308450

Eisenhardt, K., & Galunic, D. (2000). Coevolving - At last, a way to make synergies work. Harvard Business Review, 78(1), 91.

Glaser, B., (1978). Theoretical Sensitivity: Advances in the Methodology of Grounded Theory, Sociology Press, Mill Valley, CA

Glaser, B., (1992). Basics of Grounded Theory Analysis: Emergence vs Forcing, Sociology Press, Mill Valley, CA.

Glaser, B., (2004). Remodeling grounded theory. FQS 5(2):1–17

Glaser, B., 2005. The Grounded Theory Perspective III: Theoretical Coding. Sociology Press, Mill Valley, CA.

Grummitt, J., (1980). Interviewing Skills. London: Industrial Society.

Gioia, D. A., Corley, K. G., & Hamilton, A. L. (2013). Seeking Qualitative Rigor in Inductive Research: Notes on the Gioia Methodology. Organizational Research Methods, 16(1), 15–31. https://doi.org/10.1177/1094428112452151

Gupta, M., George, J. F., & Xia, W. (2019). Relationships between IT department culture and agile software development practices: An empirical investigation. International Journal of Information Management, 44, 13–24. https://doi.org/10.1016/j.ijinfomgt.2018.09.006

Highsmith, J. A. (2000). Adaptive Software Development: A Collaborative Approach to Managing Complex Systems [Article]. Computerworld, 64.

Highsmith, J. A. (2002). Agile software development ecosystems. Addison-Wesley.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. In Computer (Vol. 34, Issue 9, pp. 120–122). https://doi.org/10.1109/2.947100

Hoda, R. (2022). Socio-Technical Grounded Theory for Software Engineering. IEEE Transactions on Software Engineering, 48(10), 3808–3832. https://doi.org/10.1109/TSE.2021.3106280

Hoda, R., Babb, J., & Nørbjerg, J. (2013). Toward Learning Teams. http://agilemanifesto.

Hoda, R., & Noble, J. (2017). Becoming Agile: A Grounded Theory of Agile Transitions in Practice. Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017, 141–151. https://doi.org/10.1109/ICSE.2017.21

Hoda, R., Noble, J., & Marshall, S. (2011). The impact of inadequate customer collaboration on self-organizing Agile teams. Information and Software Technology, 53(5), 521–534. https://doi.org/10.1016/j.infsof.2010.10.009

Hoda, R., Noble, J., & Marshall, S. (2012). Developing a grounded theory to explain the practices of self-organizing Agile teams. Empirical Software Engineering, 17(6), 609–639. https://doi.org/10.1007/s10664-011-9161-0

Hoda, R., Noble, J., & Marshall, S. (2013). Self-organizing roles on agile software development teams. IEEE Transactions on Software Engineering, 39(3), 422–444. https://doi.org/10.1109/TSE.2012.30

Highsmith, J. (2004). Agile Project Management: Creating Innovative Products. Addison-Weasley.

Johnson, P. and Clark, M. (2006) 'Editors' introduction: Mapping the terrain: An overview of business and management research methodologies', in P. Johnson and M. Clark (eds) Business and Management Research Methodologies. London: Sage, pp. xxv–iv.

Katzenbach, J.R., Smith, D.K. (1993). The Discipline of teams. Harvard Business Review 71 (2) 111- 120.

Kalus, G., Kuhrmann, M. (2013). Criteria for software process tailoring: a systematic review. In: Proceedings of the 2013 International Conference on Software and System Process. New York: ACM Press; p. 171-180.

Kaufmann, C., Cock, A., & Gemünden, H.G. (2020). Emerging strategy recognition in agile portfolios. International Journal of Project Management. 38(7) 429-440.

Lindskog, C., & Netz, J. (2021). Balancing between stability and change in Agile teams. International Journal of Managing Projects in Business, 14(7), 1529–1554. https://doi.org/10.1108/IJMPB-12-2020-0366

Madampe, K., Hoda, R., & Singh, P. (2020). Towards Understanding Emotional Response to Requirements Changes in Agile Teams. Proceedings - 2020 ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER 2020, 37–40. https://doi.org/10.1145/3377816.3381722

McMullin, C. (2021). Transcription and Qualitative Methods: Implications for Third Sector Research. Voluntas, 34(1), 140–153. https://doi.org/10.1007/s11266-021-00400-3

Melo, C., Cruzes, D. S., Kon, F., & Conradi, R. (2011). Agile team perceptions of productivity factors. Proceedings - 2011 Agile Conference, Agile 2011, 57–66. https://doi.org/10.1109/AGILE.2011.35

Ngwenyama, O., & Nørbjerg, J. (2010). Software process improvement with weak management support: an analysis of the dynamics of intra-organizational alliances in IS change initiatives. European Journal of Information Systems, 19, 303–319. https://doi.org/10.1057/ejis.2010.18

Parry, K. W. (1998). GROUNDED THEORY AND SOCIAL PROCESS: A NEW DIRECTION FOR LEADERSHIP RESEARCH.

Paulk, M.C. (2001). “Extreme programming from a CMM perspective”, IEEE Software18(6) (2001) 19–26.

Robinson, O. C. (2014). Sampling in Interview-Based Qualitative Research: A Theoretical and Practical Guide. Qualitative Research in Psychology, 11(1), 25–41. https://doi.org/10.1080/14780887.2013.801543

Richardson, R., Kramer, E. H. (2006). “Abduction as the type of inference that characterizes the development of a grounded theory,” Qualitative Res., vol. 6, no. 4, pp. 497–513.

Saunders, M. N. K., Lewis, P., & Thornhill, A. (2019). Research methods for business students (8th ed.). Pearson.

Sutherland, J. (2020). Why 47% of Agile Transformations Fail. https://scruminc.wpenginepowered.com/wp-content/uploads/2020/08/Why-47-of-Agile-Transformations-Fail.pdf

Schwaber, K. (2002). “Controlled chaos: living on the edge”, http://www.agilealliance.org/articles/articles/ap.pdf.

Schwaber, K., Beedle, M. (2002). Agile Software Development with Scrum. Upper Saddle River, New Jersey: Prentice Hall. ISBN: 0130676349

Sofaer, S. (1999). Qualitative Methods: What Are They and Why Use Them?

Georgieva, S., and Allan, G. (2008) “Best practices in project management through a grounded theory lens”, Electronic Journal of Business Re-search Methods 6, 1, pp. 43–52.

Turk D., France R., Rumpe B. (2002). “Limitations of agile software processes”, in: Proc.3rd International Conference on eXtreme Programming and Agile Processes in Soft-ware Engineering—XP2002. Available: http://www4.informatik.tu-muenchen.de/~rumpe/ps/XP02.Limitations.pdf.

Vidgen, R., & Wang, X. (2009). Coevolving Systems and the Organization of Agile Software Development. Information Systems Research, 20(3), 355–376. https://doi.org/10.1287/isre.1090.0237

Yu, X., & Petter, S. (2014). Understanding agile software development practices using shared mental models theory. Information and Software Technology, 56(8), 911–921. https://doi.org/10.1016/j.infsof.2014.02.010

8. APPENDICES

Agile interview with Participant One (Speaker 1: Researcher; Speaker 2: Participant)

Speaker1: [00:00:02] Let's start with the first question. So could you please just tell me what do you think about the current work? How is it?

Speaker2: [00:00:14] Yeah. I won't mention too much about the project. Right. I think what's really nice is that when you have project requirements from the client and then we're able to sort of capture that into different specific enablers or features and then be able to further break that down into tasks for the actual developers to start working on. And then with those things, we can also then time block them into sprints of two weeks and then have all the updates available for each task on a board. This sort of setup, in general, I think it's very systematic, and it's good for tracking the progress of the project. And then, in terms of future planning, it also makes a lot of sense. So I think the general setup that I am working with or have worked with it's pretty like a systematic setup of, I think, delivering projects.

Speaker1: [00:01:32] Okay. So you mentioned the working methods like the breaking the task down, and also, you're working under the time block schedule and tracking the progress. Yeah, that's great. So, generally speaking, what do you think about the Agile method that would be the most valuable part of your opinion?

Speaker2: [00:02:00] To be very specific about the Agile method, right? Could you brief me through what is the Agile method first before I identify the most valuable part based on your question?

Speaker1: [00:02:14] Yeah. Okay. So we usually say the agile values. We focus on the individual, and we focus on the interactions instead of following the documentation. And we say we focus on communicating with the customers rather than negotiating with them based on the contract. Yeah. And also, we respond to changes instead of just following the plan. So that would be some great values of Agile.

Speaker2: [00:02:53] Yeah. So I would say for me, the most valuable one out of these is the need or rather the focus on communicating between team members and also with the stakeholders whenever it's time for either closing or planning, right? So the daily standups, for example, will highlight to us which are some of the blockers that we currently face and how they would impact our own work. So I think that is very, I think, crucial for smooth delivery of stuff. And then at the same time, you know, whenever it's time for closing or whenever it's time for planning and then when we get the sort of the client involved as well in this discussions, it's great. They know the struggles in these frames, for example, two weeks, rather than only knowing at the start and at the end. That, you know, there are some struggles, and then things might not be able to be delivered in time and stuff like that. So they actually do get a sort of more, I would say, timely. How would you say your feedback on how the project is going and stuff like that, right? So I think the part about communicating with both internal and stakeholders is really valuable for me out of this set of agile values that you talked about.

Speaker1: [00:04:29] So in your opinion, collecting timely feedback is important for you?

Speaker2: [00:04:38] Right. At least for me to know how it impacts my work. And at the same time, I guess for the client, maybe it also helps them know that because of some blockers that we have at the

moment, we're not able to deliver the project because so and so. So then I think these kinds of communication will enable like a really, I guess, smooth working and like a working condition. So, I think this is at least the most valuable for me.

Speaker1: [00:05:10] Okay. And regarding responding to changes, have you ever encountered any requirement change from stakeholders or customers?

Speaker2: [00:05:22] Not really, because I think the project that we have now is pretty straightforward, like what should be delivered. And there's not really going to be any deviation from that because the task is, at the core of it, pretty straightforward. So it's not going to change what is required. It's just how long that's going to take because there are a lot of dependencies in between. It depends on other organizations and whenever they deliver before we can actually start work. Right? So, um, in, in terms of that. I have never really faced much of a change in requirement based on what we are supposed to deliver as agreed on day one. But yeah, the change that I have is more like the timeline rather than the requirements. At least that's. That's my experience with the couple of projects that I have.

Speaker1: [00:06:30] Okay. So you mentioned the dependencies that will actually influence your workflow. Sometimes you will be blocked by the other dependencies. Yes, I agree with you. And regarding this problem, like you have a lot of dependencies, have you ever learned or summarized any experiences on how to figure out these kinds of issues in the future? Or do you think, as a whole team, you try some kinds of ways to solve this issue?

Speaker2: [00:07:11] So the learnings, are you talking about predicting dependencies? Because in this case, it would mean that running the same project with the old information of knowing who is supposed to wait for who. And then, if we already know that, we should note that down. And then if we have to run this project again, then we will just refer back to our old enablers or features. To figure out who we were waiting for and who we had to wait for. Then we have to prioritize working on those first, right? So yeah, that would be the learning that I would imagine. But this is only very specific to projects where you will run the same thing again in a different environment, right? But for other projects, it's really hard to, I guess, predict dependencies until you're actually in one. So learning in that regard, I don't know if I have anything specific, but I guess it really still boils down to expertise and communication. So if somebody knows that whatever they are going to be assigned has a dependency on something else, then it really counts on them to bring that up during planning meetings or during the standups. So it depends on the developers' sort of own foresight to communicate that during a meeting, regardless of whether it's a daily standup or planning [00:08:54]. So I would say that it takes a little bit of that to really work on dependencies, but otherwise, you only kind of really know when you're in it. And at that point, it might be a bit too late, but I think that's just what it is, right? And then you sort of adapt to changes as per your other agile value. [00:09:18]

Speaker1: [00:09:19] Why I asked this question because in the Agile manifesto, there is some kind of value or practice we call self-learning or inspect and adapt. So this process would be the whole team should be a self-organizing and self-learning team. So I would like to understand what you think of these kinds of self-learning activities in the team. Do you think it's important, or how do you implement this self-learning, or how do you plan to do it better in the future?

Speaker2: [00:10:11] The first question was again?

Speaker1: [00:10:15] How do you think the self-learning thing, do you think it's important for the team to develop?

Speaker2: [00:10:27] Yeah, I definitely think that's important for every team member to be self-learning in order to deliver the project. Because you know, when we come back to like talking about having foresight on dependencies when you work on something. That in itself means that if you have that foresight, it means that you have done that self-learning. But I think it's very important to note that as well that not everybody in the project would necessarily have the expertise or the time to really build up on that knowledge in which they start having foresight on dependencies. So while I think self-learning is important because the Agile sprints are so short, sometimes you just have to take an enabler and run with it and then sort of deliver within that two weeks. You don't really have time to sit on that and then think really hard about the things that could be around it. So while I think the concept is great and people should definitely do it, when it comes down to the reality of delivering an enabler or delivering something within the sprints of two weeks, it might be a little bit difficult for people to learn or self learn and at the same time deliver.

Speaker2: [00:12:02] So sometimes, people will prioritize delivery over self-learning. Then this becomes that they just trying to meet the acceptance criteria of a specific enabler without really knowing the implications of if something that they have done wasn't so correct, what would that lead to? For example, this is it's just a balance between time and expertise in this case. So my point is it's important, yes, cool, that people do self-learning. But the reality is we have a timeline that we have to meet. And that degree of self-learning can differ from person to person depending on how much priority they put in and how much effort they put in as well.

Speaker1: [00:13:01] Yes, because we have this kind of retrospective meeting, and during the meeting, we discuss what went well and what didn't go well, and what we need to improve. So we have some kind of improvement items and implementing them in the next iteration. So we can say this is some kind of self-learning. But also, you mentioned a very valuable point that there is some kind of balance between the time pressure and this kind of expertise. I agree with you. So you have this kind of pressure. You mentioned that you would try to commit to trying to deliver the enablers instead of improving your work. Do you think it's a good thing, or do you think that's something we need to improve? Like we spend some extra time just focusing on improving our expertise?

Speaker2: [00:14:16] I think okay, so it also depends on the person, right? On the developer, on the team. Because when we have this sort of timely sprints, it prevents people from being perfectionists. I think that is in itself a good thing because when someone, when a team tries to be perfect, you really don't then achieve a lot of deliveries. Because you can never be perfect. Right. So then your delivery will be very perfect, but you don't deliver much because it's just impossible. So I think the balance. So I think the sprint thing will mitigate that in the sense you prevent people from trying to be perfect in their solutions. But at the same time, I believe we also should learn from what we do in the limited time before. So having improvement items at the end of a project where we take the time, if available, to sort of rework some of the more critical stuff that we have done before. I think this is sort of a good approach because when you do that, people do get a fresh pair of eyes on some of the old things that they have done. And this then gives them a good idea. Or sometimes, when you look at things with fresh eyes, you might think of something that you didn't think about before, and then this will really help you and improve the work that you have done already. And in this sense, improvement to me is a lot more

valuable than someone trying to be or a team trying to be perfect from day one. So I think that this to me personally I would prefer this approach where it's an iterative approach. We do something, it may not be perfect, but we deliver. And then at the end of it, if we do have some time, we come back to it again, and then we sort of improve that if we can. And then and then we deliver like a second or version two of what was delivered. So I do prefer this approach.

Speaker1: [00:16:37] But do you think so far in your current project you have already got enough time for your self-learning or not?

Speaker2: [00:17:00] Yeah, I think I do just in general.

Speaker1: [00:17:09] Yeah. Okay. So maybe, uh, could you just give a rough guesstimation on how many percent of your work or working hours would be put into this kind of improvement work item? Do you think it's good, or is it enough for you to do the self-learning?

Speaker2: [00:17:32] Because I would consider myself learning, even though I'm not just assigned, even though I'm just working on an enabler. I don't necessarily have to go into the improvement stage to be learning. I think that the learning already takes place when I am working on an enabler from day one. So in that sense, the amount of time that goes into learning versus actual development, I would say maybe 30% is learning 70% of that time in the enabler is delivering or developing. Okay. But when we go to the improvement stage, where we come back to an old enabler and then we try and work on something, the learning there would take a bigger percentage. Because I can already see what I have done, and if I want to make it even better, this requires more Googling or more research and stuff like that. So the learning part in that phase would take a swap. So it will be 70% learning now and then 30% actually improving on what I have done or writing a new set of codes, for example.

Speaker1: [00:18:54] Yeah. Okay. And so regarding this kind of learning effort you put in your work, I'd like to know when you estimate your workload. Will you count these kinds of learning time into your estimation, or it's not included?

Speaker2: [00:19:27] So when I take an enabler, if I don't fully understand what it is, I would take into account the time that I have to Google and I have to research before I'm able to deliver into the total time that I would take to deliver this enabler. So I would take the whole. I would take both.

Speaker1: [00:19:48] Okay. And according to your understanding so far, and the current project that you're working on. Do you think it's a self-organizing team? Can everybody take autonomy by themselves?

Speaker2: [00:20:04] To an extent, because we do have the autonomy to decide how we want to deliver the work. But there is also going to be a need for a pull request for the work to be published. And when we do the pull request, it's generally the product owner or the solution owner who is going to give you feedback and ask you for more things out of your pull request. So when that happens, it means a lot of the work that we are going to publish is based on what the solution owner or product owner requires. So in this sense, we do have some autonomy at our level. But when it comes to actually getting it accepted, there is still a standard to meet. Before it gets accepted, so in this sense, I would say we have a good amount of autonomy, but there's still some extent of having to meet someone else's criteria before it gets accepted.

Speaker1: [00:21:34] I agree with you. Actually, we are developing something to meet the standard of the GxP. So that would be different from developing the mobile application or something. There are some balances we need to meet. But yes, I agree with you. Okay. I think we still have five minutes. I would like to ask the last question. Regarding these kinds of agile activities or agile values, do you have any other comments you'd like to mention here?

Speaker2: [00:22:22] Uh, no, not really. I think the setup is good. But it also depends on how a team to properly work agile. It's very crucial that every team member knows the methodology and knows what is expected of them for this to really work fine. Because if somebody doesn't know what an enabler is, somebody doesn't know what acceptance criteria are, then it becomes a little bit tricky for them to get used to this methodology. And if one person doesn't follow it, it can trickle down into the work of other people. Because as we talked about, there are dependencies on people's work. So it's important that everybody is on the same page about working agile. And then and I think this could be actually a really tough part of actually working agile, just getting people on the same page. Yeah.

Speaker2: [00:23:32] So yeah, but otherwise. The framework in itself is pretty straightforward to understand. And if everybody is on board then it should be fine too.

Speaker1: [00:23:47] Yeah. It's actually a simple structure like the scrum process. But when you implement it in your projects, you will encounter a lot of problems because you have the project context issue. So that definitely should be adapted according to the different project contexts. So that's why I think the Agile value is advocating this kind of responding to changes. So everybody should learn, should try to figure out the question, the solutions, and whatever they think they should do to improve their whole process. I totally agree with you. Thank you so much. That's really helpful and valuable input for my master's thesis. Thank you so much.

Agile interview with Participant Two (Speaker 1: Researcher; Speaker 2: Participant)

Speaker1: [00:00:00] Okay, great. How's your project? Could you please just explain a little bit regarding it?

Speaker2: [00:00:10] So. Well, my. My project is. Is. Well, can I talk?

Speaker1: [00:00:15] No, not the name. Just something the agile way. How you follow the process.

Speaker2: [00:00:21] We are a small project. Yeah. So we are only two developers or three developers and sort of a PO. So we have light agile in the sense that it doesn't make sense to roll out the full thing because coordinating between four people and the total is a very easy job. so we do have dailies. We are struggling a bit with finding the right balance between sort of having all the official meetings, like what they call iteration planning, and all those reviews and all.

Speaker1: [00:01:00] Yeah. And the retrospective. Yes as well. Do you do that? Okay. No, No. Retrospective.

Speaker2: [00:01:05] No, not at the moment.

Speaker1: [00:01:06] Okay.

Speaker1: [00:01:07] So um, usually which, which kind of activities. I mean the agile activities. Do you think is the most important or most efficient or effective, or valuable in your opinion?

Speaker2: [00:01:25] I think the Daily has something to it because you get a kind of idea and usually you have it in the morning. You can kind of figure out, okay, who needs help so we can all move forward during the day. Um, right now, we are also doing it with an international guy who is not sitting in our time zone. So we have it moved a bit, which is sort of reducing the value of the, daily because, you know, we are already halfway more than halfway through our day before we have our daily, which makes it a bit (less valuable).

Speaker1: [00:01:55] Yeah

Speaker2: [00:01:57] Otherwise, I think the iteration planning and sort of setting the goal of the next iteration. Yes. We want to achieve this, this, and this. Yeah. Yes.

Speaker1: [00:02:07] It helps with some kind of alignment. Information alignment would be important.

Speaker2: [00:02:11] So I think most of the alignment from the customer to say, okay, what I would like, you know what my priorities are this, this and this. And then we can start sort of figuring out, okay, what can we achieve? Of all your priorities, what can we achieve?

Speaker1: [00:02:27] So will you interact with your customers directly? Okay. Your customer will join this kind of planning meeting. Yeah.

Speaker2: [00:02:37] Po is our customer. So he embodies the whole thing to say.

Speaker1: [00:02:42] Okay, so your PO will prioritize all these work items. But will your PO change the scope from time to time? Yes. Okay. So will that influence your work efficiency? I mean that during one iteration will your PO change?

Speaker2: [00:03:08] Yes, yes.

Speaker1: [00:03:09] Okay. How do you feel about it?

Speaker2: [00:03:12] Um, I think it mainly comes from the fact that we don't have an operations team. Okay. So agile is, to me very geared towards creating new. Yeah, but we are so few people that, of course, if bugs come in or the application goes down for one way or another, there's nothing sort of shielding us from that. So, you know, while we are doing new features, then we can be pulled out because now this bug, then we need to look at the bug immediately because we are both in development but we also in operations.

Speaker1: [00:03:42] Okay. Before you implemented this kind of agile method, how did you work there?

Speaker2: [00:03:52] We had always had Agile in that project.

Speaker1: [00:03:56] Okay. Always have the Agile.

Speaker2: [00:03:59] Not fully developed but always a sort of work concept. Yeah.

Speaker1: [00:04:02] But when you figure out you have some conflicts, why are you still working on the development and operation at the same time? Uh, have you ever tried to figure out the way to solve this or balance this work?

Speaker2: [00:04:20] I have in another project, but if this is specific to it?

Speaker1: [00:04:25] You can just talk about whatever.

Speaker2: [00:04:27] So I had another project with sort of, a bigger team. And the way we sort of solved it was that this was also this project was more heavily engaged in Agile, than the first project. Yeah. And the way we handled it there was that we set aside, set let's say 10% of our capacity, was set aside for bugs. So when we did iteration planning, we had a story called Bugs. There was something healing, we would put a task under that and then transfer some of the time from the feature user story down to the task. And once the sort of feature ran out of capacity. It would sort of not do more bugs for that week because, you know, if you say, I think we used hours at that time, we said we put ten hours aside, and if something comes in, then you know this bug, or it took me two hours to fix, I take two hours out of that pool, and then that's eight hours left for the rest of the sprint for people to do something.

Speaker1: [00:05:38] So how did you figure out this kind of 10% time?

Speaker2: [00:05:43] That was just a sort of let's try out. So we tried out. We never sort of adjusted it, but it fits quite well with I, I don't remember if it was ten hours, but you know, we tried different things trying to sort of have bugs and then just wait until the next iteration to sort of plan them in. But that was too late and some of the bugs had to be fixed right away. This kind of course, we didn't delay our deliveries because we knew we had some time in our sprint to handle bugs, which was great.

Speaker1: [00:06:17] Yeah. Yeah. Okay. I'm very interested in how you really figure out, or, on which occasion you try to figure out or try out this kind of 10% bug fixing time.

Speaker2: [00:06:40] Like, it came out of the fact that you know, we were new as a team. We were still figuring out how much capacity the team had. And for some time, we always overestimate how much we could finish. And it was always the question of, well, I was doing this, this, and this bug, so I couldn't work on mine. And you know, we attempted to, we had many ideas. But what really worked was that we set time aside and then we looked at each other like we tried to see how much. Because how much time have we spent on bugs previously to see what was actually sort of the baseline for the bugs? So we registered all bugs and sort of put the time on them that it took. But it was only later that we actually came up with the idea to reserve time for it.

Speaker1: [00:07:31] Yes. But do you have any specific meeting to discuss this kind of improvement?

Speaker2: [00:07:39] That usually happens during retrospectives.

Speaker1: [00:07:41] Okay. Do you do retrospectives?

Speaker2: [00:07:42] Yeah, we did. In the second project, I talked about that. Then we did sort of all the Agile meetings. It was very sort of by the book, Agile.

Speaker1: [00:07:50] By the book, yeah. But do you think that kind of iterative meeting or retrospective meeting is important or useful?

Speaker2: [00:07:59] Uh, I think we had a scrum master that was sometimes hijacking it for other purposes which made the used it for learning in general about Agile, which reduced the value of it to me. So some because it became very sort of teaching sessions which had little value in the end. And it prevented us from actually doing retrospectives which we at some point needed but didn't have the time for it. And then we kept having bad habits for maybe an iteration more.

Speaker1: [00:08:31] So in your opinion, what kind of retrospective topic would be valuable for the team?

Speaker2: [00:08:39] I think it would be just the general retrospective without sort of a fixed agenda because it is about. What went wrong, Why didn't we finish in time and what can we do to improve it? And so I think sometimes the retrospective can be five-ten minutes, sometimes it can be half an hour, an hour because you need to discuss things.

Speaker1: [00:09:02] Yeah.

Speaker2: [00:09:02] Yeah. But I think it's important to have that flexibility in that meeting.

Speaker1: [00:09:08] Okay. But after the retrospective meeting will your team really implement that improvement or solution into the next iteration?

Speaker2: [00:09:19] One of them was this allocating bug time that was out of the retrospective as far as I remember.

Speaker1: [00:09:26] So you spent some time on the improvement work item, something like, you spend 10% of your time fixing bugs. But were these improvement work items counted in that 10% bug fixing time or it's another part?

Speaker2: [00:09:50] It's another part.

Speaker1: [00:09:51] Okay. Yeah.

Speaker1: [00:09:54] That's great. But as a whole team, do you think that you can cooperate with each other to respond to the challenges of the customers?

Speaker2: [00:10:11] Yes.

Speaker1: [00:10:14] Or as a team member, will you be influenced or affected by this kind of too frequent changes from the other parts?

Speaker2: [00:10:28] I think in my first project, number one, it's we're getting sort of it's challenging to run the Agile when we're so few and we get so much noise from the side that we don't have a fixed, we can't have a fixed scope, because we will keep getting in the one-second project. It was quite easy to follow because it was limited. Of course, once in a while that got in something that shouldn't. But in general, we are close to the scope and then we work towards that. So, um, the difference was also, you know, the first project is, was an application that was running in production. The second one was still not in use. So that also changes something.

Speaker1: [00:11:17] Do you think that just following the agile process by book is more efficient or do you think that makes adaptation for the project will be more efficient?

Speaker2: [00:11:28] I think it has to be adapted to the team. Yeah. Uh, also, since, you know, sometimes people are not fully allocated, agile can quickly if you as an Agile requires you to be fully allocated to a project to really make value because otherwise those who are not well spending all their time in scheduled meetings and not actually doing something or not have any time for doing something.

Speaker1: [00:11:57] Doing something means that these things are related to the features or anything, or these things are related to your self-learning continuous improvement or both?

Speaker2: [00:12:14] I mean mostly towards the sort of if you're, let's say 50% allocated to the project and runs Agile, then you will see 50 to 80% of your time being lost to meetings from Agile, which all of them. But that also means that you can only contribute 20% of your time. So you end up not providing that much value. But if you're fully allocated then you know it's a smaller part of your hours. So yeah, I think it's also some of the agile meetings we sort of by the book require everyone in the team to be there. Yeah. And I would maybe from what I saw was that maybe you only need sort of the, the more senior people of the team to sort of can reduce the scope at least. So if it's normally one hour with the team make it half an hour with you know only the senior part and then one hour the other half an hour with the whole people.

Speaker1: [00:13:14] Yeah, exactly.

Speaker2: [00:13:15] So we don't sit, and you know have because you quickly end up spending wasting people's time if you sit and have heavy architecture discussions where not everyone is relevant.

Speaker1: [00:13:28] Yes. And we know that we have some different kinds of agile values like responding to changes. We respect individuals and interact with each other, face to face, and we focus on the working software. In your opinion, which one, which kind of agile value you think would be most precious for you or for the team to develop?

Speaker2: [00:13:59] Can you, can you rephrase that one?

Speaker1: [00:14:01] I mean the most important agile value that you think will really help the team to develop.

Speaker2: [00:14:10] I think it is to cut it down. I don't know if it's the value, but it's the fact that you cut it down into very small and tangible, sort of the fact that you boil it down to small value-adding things so you don't try and start to build a whole application, but you keep it very sort of customer focused. Why are we even doing this? It's because it creates this little part of the value. Yeah, I think that's sort of the biggest driver of value from Agile and also that you can sort of yeah, that you can test out if you're even going in the right direction without committing a ton of resources. Yeah.

Speaker1: [00:14:47] If you really break them down, break this task down into smaller ones. And after the iteration, when you figure out you have done something wrong or anything you need to improve, you make improvements in the next iteration. I see that breaking them down as one part and also getting the output from the first iteration and putting them into the next iteration. That kind of self-learning is also very important.

Speaker2: [00:15:23] Yes. And I think it also helps in, Yeah. Aligning expectations because you maybe believe you can do it all in ten months and then one month in you can sort of see that it's not you haven't committed to this long term because.

Speaker1: [00:15:38] But will you be pressured by your product or customers, like you have to deliver something by end of the period of time?

Speaker2: [00:15:46] No, I think in general, that's sort of the product owner the scrum master's responsibility to shield the developers from that part. So you know, if they stay true to the Agile, then the team promises for this iteration, we will deliver this. But you know, the long-term scope is not something you can promise and not something you should at least ask the whole team to be bothered with.

Speaker1: [00:16:13] Yeah. But will you usually fully deliver your commitments after one iteration?

Speaker2: [00:16:23] No.

Speaker1: [00:16:23] Okay. But what will you do then? What will you do if you cannot complete all the tasks or commitments?

Speaker1: [00:16:35] Will you discuss how to improve that?

Speaker2: [00:16:37] Yeah, we take that in a retrospective. For one project we discussed actually extending, making the iteration bigger because we felt that the Agile meetings made sense, but they took up too much of our time. So instead we wanted to extend by one week or so we would have more time in between the meetings. Yeah.

Speaker1: [00:17:02] So you modify your team development cadence based on the retrospective output?

Speaker1: [00:17:10] Yes. Yeah.

Speaker5: [00:17:11] That would be perfect.

Speaker1: [00:17:12] Yes.

Speaker1: [00:17:14] Okay. And maybe the last question. Do you have anything that you want to talk about, regarding what you want to keep on continuing doing in the future? You hope that agile, this kind of self-improvement or self-learning team can bring you some more valuable outputs.

Speaker2: [00:17:51] I don't know. I think Agile struggles a bit in the classical world of what we work in because most of the organization is the waterfall, and then you try to put Agile into it, which quickly becomes a challenge.

Speaker1: [00:18:06] And also we're currently trying to combine Agile with DevOps, so maybe something Pipeline. We can't combine them together to improve working efficiency (in the traditional way).

Speaker2: [00:18:26] I think the core aspect is to have people that you need or want to go live with that product and you still want to do Agile. Then you need a support team that can in the beginning, when you have a brand new project, it's difficult to do Agile in maybe the first sets because you need some infrastructure to be in place and you know.

Speaker1: [00:18:50] Yeah.

Speaker2: [00:18:51] Sort of before you can start making small chunks of value, you sort of have to put the foundation before you can start putting a wall that and a wall. But the foundation takes the amount of time it takes. So I don't know if Agile works that well in the beginning. Mostly because some things don't create value. It's just the foundation before you can actually start creating value.

Speaker1: [00:19:15] Yeah, it is some kind of thing that we need to make the definition of the ready for this kind of thing. Definition of ready for implementing the Agile.

Speaker2: [00:19:26] Yes, exactly. Yeah. Yeah.

Speaker1: [00:19:29] That's great. Yeah. And I think that's almost a half hour. Thank you.

Agile interview with Participant Three (Speaker 1: Researcher: Speaker 2: Participant)

Speaker1: [00:00:01] Great. Thank you. From the very beginning, how is your project running?

Speaker2: [00:00:06] My project has been running well since we implemented the Agile sprints. Yeah, the sprint sessions with the standards and Sprint closing, actually. Yeah. It went very well. Okay. I think that for every one of the members, it was very good to have an overview of what the others were doing, and they could understand better why their tasks influenced the others and so on. It was also all the standards that we are having is a very good meeting point for saying, okay, I'm doing this task, but I need your input. So let's meet half an hour later or during the day and so on. So the collaboration has improved a lot. And basically, the better understand the different members of the overall project, and we are going to test this week the Sprint review, where we are going to create a PowerPoint of four slides, one per team member, and then send it to the product owner. And we are going to present it this week to ourselves, but maybe next week we will do it with the product owner also.

Speaker1: [00:01:25] Okay. So your product owner is your customer.

Speaker2: [00:01:30] Not really. It's both. Yeah. Product owner and customer. Yes.

Speaker1: [00:01:36] Yeah. So far, you have already implemented the main structure. Yeah, the main structure, uh, daily standup, sprint planning, planning, and Retrospective.

Speaker2: [00:01:48] Okay. Sprint closing. Yeah. And retrospective with it together. And then Sprint review we are going to implement this week. And refinement. We are going to start this week also on Wednesday.

Speaker2: [00:02:00] So we are trying to build all the blocks progressively. Not yeah, not all at once because we don't have a scrum master. So we are trying to adapt it to our way of working. But I think it improved for sure the collaboration and the understanding of the project. Yes.

Speaker1: [00:02:20] Yes exactly. So it seems that you are a very good self-organizing team since, without a scrum master, you did well.

Speaker2: [00:02:28] Well, because we are a small team, we are five. So it's fairly easy to organize. If we were more, maybe we couldn't. Yeah. And each of us is very proactive in knowing what kind of tasks we should lead and be the leaders on and then how we can get help from others. So the team also makes it

easy to work without a scrum master. Okay. Of course, I would love to have a scrum master. Maybe XXX is going to become our scrum master at some point. But I was worried because I feel like I don't want to spend much time on those kinds of tasks in a scrum master because then I lose focus on my other tasks. Yeah. So I prefer someone specialized in it, but it's working well in our way. We are not the best agile team, but we are organized. Yeah, I think that's good.

Speaker1: [00:03:26] Yeah. And also you mentioned the people. The team members are very proactive. Yeah. Could you please elaborate more on how they do?

Speaker2: [00:03:34] So we have very clear backgrounds. I would say I'm the most experienced in data engineering. Then we have XXX, the UX lead. And then we have XXX, who are the top AWS programmers and developers. And then we have XXX from the data science side. And I think each of us understands our role and how we can get help from others. For example, I know that as a lead engineer, I need to interact a lot with XXX, but I need engineers to understand what we are doing so then they can be proactive. And they weren't proactive before starting the Agile method working in Sprint. But now I can feel like they understand better what we are doing, and they propose more solutions that are for me proactive because they propose solutions, meaning that they are thinking about the solution and working towards a common goal.

Speaker2: [00:04:49] Otherwise, they understand more. They would just limit themselves to solving the tasks that I delegate to them. But now they are becoming better at thinking about the overall solution.

Speaker1: [00:05:05] Yeah, that's good.

Speaker2: [00:05:05] XXX has done it since the beginning, more or less. But it's difficult for India because we have a difference in time timezone. We cannot meet physically, so we need to improve this communication, and I think the agile planning and the sprints have helped a lot. Also, the structure for having standups, even though it's 15 minutes, gives a clear idea of if they are struggling or not. And they take those 15 minutes to ask for help. Yeah. So I think, I think it's pretty good.

Speaker1: [00:05:44] Yeah, that's good. And you mentioned that they understand better and they are also changing a lot. So could you please elaborate on that? Yeah. How did they make these changes?

Speaker2: [00:05:57] How are we all improved as a team? As a team? Yeah. Yeah. It's more as a team that we improve. Yeah, for example, in the standups, we go through user story by user story or an enabler by an enabler. Yeah. So. We make sure to explain. We made sure that we explained every user story in a very plain language, so the developers could understand what the UX was explaining, and at the same time, we wanted the UX because it's an extreme UX developer. So from the UX side, XXX can understand A and B without the developers. And that makes XXX realize what they are working on and what is the influence of their work on the overall project? Yeah, the same. What we try to communicate is to to be mentioned that they are not just working in they are not just delivering tasks. They are working towards a common goal. So they have to be more creative than just writing code. And I think they understand this. I think everything as a whole is working towards [00:07:33] a better understanding of the common goal. [00:07:34]

Speaker1: [00:07:35] What did you mean by the common goal?

Speaker2: [00:07:39] The common goal is the goal of the project. So the different epics we are working on, we have an epic called data pipeline, so we are working on infrastructure for this data pipeline. Then we have the data visualization where the UX has more part of it, but most part of it is UX. But we are working mainly in the data pipeline, for example. But there is another goal, which is data visualization. Yeah, but data pipeline affects data visualization, they need to understand how this data pipeline epic affects the other epic. And I think they have a clearer view now of what was happening before now before. We explained it in some sessions, but we didn't have the regularity that makes you follow, you know, makes you follow day by day what is happening. And I think that that's what makes them understand now what's happening.

Speaker1: [00:08:49] So do you think that this kind of agile process helps them to improve and understand the project?

Speaker2: [00:08:57] Be not just delivery, but be creative also in their solutions.

Speaker2: [00:09:06] Okay. So in development, you can just code and solve a problem or you can do better, you can solve the problem and challenge it. So, for example, they have to write some code that does something specific, but I don't know, it's copy file from A to E. You can write so many different kinds of code to that. And you will deliver the task it copies A to. But what happens depending on how you copy? You will influence the UX data visualization, for example. I don't know. So when they receive the task, they should challenge the requirement by saying, okay, if we copy a B in this way, this is going to happen. If we copy a B in this way, this is going to happen. And then. The UX and I can meet and say, okay, we have these two options that demonstrate a suggested, which one we should take. This one. Okay. If this didn't happen, they would have taken one of them without asking. That's not the way. So because we have to collaborate, they should propose back, which are the two solutions that they have found. And then, we decide all together have a meeting maybe or whatever. And this regularity in the sprints makes us be better at asking the right questions to each other. That's how I feel it.

Speaker1: [00:10:49] Okay. So before they didn't do that.

Speaker2: [00:10:51] Before it was more difficult. It was more difficult because we were not meeting so often. Okay. Yeah. And it's not a long meeting. We don't need a long meeting. We need just 15 minutes. Just stand up. That was I think it's working very well. And the sprint closing because in the sprint closing, we go through each story that we have to close. Yeah. So this means that the developers will see the UX tasks and UX will see the developers' tasks. So I think in this sense in the collaborative understanding of the common solution, it's working well.

Speaker1: [00:11:28] Yeah. Okay. So how many sprints have you taken?

Speaker2: [00:11:33] Three.

Speaker1: [00:11:34] Three Okay. After these three iterations or sprints, what kind of lessons you think that the team learned?

Speaker2: [00:11:46] Yeah. So one of the lessons is the team is working towards a common goal. So you need to communicate. Yeah, [00:12:04] everyone needs to understand what are the tasks from [00:12:07] the others. That's one of the lessons. Lesson two. You don't need long meetings for doing that

Speaker1: [00:12:17] You need or you don't need long meetings?

Speaker2: [00:12:20] You don't need long meetings to understand what are the others doing or what is the common goal. You don't need that.

Speaker1: [00:12:28] But how did you do that? I mean, you don't need that kind of long meeting, but you you split them into the Daily standup meetings.

Speaker2: [00:12:43] So it's better standup than a long meeting. Talking about a lot of stuff that at some point you get bored about listening from.

Speaker2: [00:12:57] Because if we have a long meeting of 1.5 hours, which everyone is explaining all what they have done and so on, yeah, you will forget after. Yeah, you lose concentration, you are going to forget the next day. It's better this long meeting to have it in small doses throughout the week in the standups and maybe then in the sprint closing or sprint review, we can have a look at one of them. That's it. Or if you need more information. We can say. Okay. Could you please make a plan, for example, mapped out some processes. XXX, we don't understand this, can we have a meeting, the both of us, to understand? And then they have the meeting and the others keep working something else. Yes. So it allows this interaction that saying, okay, let's have a meeting apart from the stand up, because we need to work together to solve some something. So that's another question. I would say yes. And the third lesson. Having regular meetings allows better communication and then it yields to the possibility of opening new meetings. You can have more meetings between those ones.

Speaker1: [00:14:55] And how do you think about the retrospective meeting? Is it valuable for you?

Speaker2: [00:15:06] Yes, I would say that we shouldn't. It's the part that I like the least. And it's not because I feel it is unuseful, but I think you can do it much faster. In general, it can be done much faster. So take the the main points like saying, okay, we should have been better at estimating how many tasks, how many user stories we can complete. Okay. The the sprint closing is taking too long of in the sprint review. In sprint review, we didn't know what to show. But. It has to be more agile. Yeah, the retrospective. Sometimes something that we get stuck. I think we have to do it more agile or. Also, I think for what I've seen in other projects also, it can be done using the word yes.

Speaker1: [00:16:10] But I think after three iterations, you actually made a lot of improvement.

Speaker2: [00:16:17] No. Yeah, I mean, from having nothing to have a structure.

Speaker1: [00:16:21] Yeah. So far, regarding the Agile thing. What would be the most important for you when you implement it into your project?

Speaker2: [00:16:35] The most important is the meeting structure. So the the structure of having a stand up, a sprint planning a closing defined time frames, that's what I value the most.

Speaker1: [00:16:56] But do you think that following the whole structure by book is good?

Speaker2: [00:17:02] No, not. Not necessarily. Not necessarily. It can be adapted to your team because we are not following the structure by the book. We're just taking what it can work for us because we are a smaller team. And by taking those and adapting it to our self. Of course, right now the structure is not the final one. So we are still adapting it. For example, because we don't have a standup every day, but maybe we are going to include one more between two days that we think there is more workload. I

don't know. So but so far the the structure of the different meetings, and the time frame, I think it is the most impactful one in our way of working.

Speaker1: [00:18:08] Okay, So it sounds like that responding to changes is something that you are doing right now. You adapted the whole structure. The team members learn from each other from each iteration.

Speaker2: [00:18:26] Yeah. I think all of us have already worked with Sprint before. No, we kind of already knew how to tackle the different meetings. We were a bit of experienced. Let's say the developers had already worked in this kind of framework and so we knew more or less. And I think we are.

Speaker1: [00:18:54] Yeah, yeah. Okay.

Speaker2: [00:18:56] But we don't have a scrum Master. If we have one, that would be very nice.

Speaker1: [00:18:59] You have done so far. Great job I think. Yeah, that's good. You set up the whole structure by yourself?

Speaker2: [00:19:06] Yeah. Also, it takes time to create the epics, the user stories, because that's very important also to have a tool that allows you to.

Speaker1: [00:19:18] Yeah. I'm very interested in your team members. They are very proactive. How did they achieve that? I mean it's not an easy thing to have this kind of characteristics from the very beginning.

Speaker2: [00:19:37] No, no. We didn't have them like that. I'm starting to see these improvements. But we can improve even more. Be more proactive. We are improving. So that's what I feel, that we are becoming more proactive and so on.

Speaker1: [00:20:04] So do you feel better or feel confident after implementing the Agile?

Speaker2: [00:20:10] Better. Better. Much better.

Speaker1: [00:20:12] Why? Why do you think so?

Speaker2: [00:20:13] We are more structured. First in the team. Second. We understand better the goals and who is doing who is the lead on what and how we can work together. The second structure, common understanding. And we will see in the coming weeks. Maybe we can have a short session of five minutes. How good are we at summarizing what we have done during the week. Because we are implementing this spring review now. I cannot tell you much about refinement. That is something we need to improve. But Sprint review is something I'm curious about, seeing how good we are in summing up what has happened during the week. Being sharp, straight to the point. And building, creating a document, some slides that are valuable for the product owner at a glance see what happened during the week.

Speaker1: [00:21:45] Yeah that'd be great. I remember you haven't ever mentioned one issue that your team member completed the task in advance.

Speaker2: [00:21:55] Yeah. Yeah, we have to improve that.

Speaker1: [00:21:57] How did you improve that?

Speaker2: [00:22:01] So I followed your advice, being better at documenting. But actually. What if instead of documenting more, what we did is looking for what can be improved of what we already have. And this created new user stories for the next sprint. So. It was not working in a new user story. It was more, okay, let's review the whole pipeline again. We are reading a pipeline. Let's review the infrastructure again. And we found out more issues that became user stories. That's what we have been doing because it's a project where new things come up all the time. So you can spend some time in reviewing and then something comes up. Maybe in the future we will not have this problem because we will be better at estimating.

Speaker1: [00:23:04] Yeah, exactly.

Speaker2: [00:23:05] But then we will have to spend this time of reviewing some. We will have to create our story for reviewing. It's just [00:23:15] some kind of learning process. You learn by yourselves step by step. [00:23:19] Yeah. that'll be great. We are still immature. It's working for us. I see a clearly change, but we are still immature, I would say.

Speaker2: [00:23:33] But I'm super positive. Yes, I'm very happy. But still, I tell you, we are mature.

Speaker1: [00:23:43] I think you are very good at self-learning, improving, changing and adapting. Yeah. one more question about responding to changes. Have you ever encountered the problem that your product owner will pressure you to deliver something by the end of specific time, so that you don't have time to work on improvement?

Speaker2: [00:24:17] Not really. The the product owner is setting the goals. We discussed with him the priorities. I know we are delivering without pressure. I would say, for now. I don't know in the future. But for now we are living without pressure because our product owner understands that this is something new with what we are building and he knows all the issues we can encounter during the process and these issues. Will delay our deliveries. And every time that we encounter new ones, we communicate to him. And he says, okay, we didn't know this. We didn't know that all these blockers, all these issues, something we had discovered. And then he understands that it's going to take longer to deliver. So he's not setting up tight deadlines. No.

Speaker1: [00:25:26] Yes. Okay. So do you think product owner's support will play a big role?

Speaker2: [00:25:32] Yeah. So far, yes. So far he he's very good at understanding the all the issues we can encounter. So. Okay. Very comprehensive in this sense he understands he's very active in understanding what we're working on. Well, he is not a developer. He's not a super tech savvy, but he reads a lot about what we use as a technology the development coding, difference of words.

Speaker1: [00:26:19] It sounds like that the whole Agile team, including your customers, should keep this kind of agile thinking so that you can collaborate more effectively.

Speaker2: [00:26:31] Yeah. we would like at some point for collaboration for sure. And we would like to include sometimes the users and we are struggling a bit on that because they don't have much time for us. So we would like to include them more.

Speaker1: [00:26:49] Yes, because you actually made some changes during the iteration because sometimes you have some hold on items, block items. So you get back to your owner. They actually agree and also accept this kind of changes. So both sides would reach this kind of agreement to make changes together. Yeah. Yeah. Accept the changes, respond to them.

Speaker2: [00:27:16] Yeah, definitely. Yeah.

Speaker1: [00:27:19] Okay. So the last question, what kind of agile value will you keep on promoting or making sure that your team members need to follow?

Speaker2: [00:27:36] I love again. I've seen this is game changer. Everyone understands what we are doing. Every team member, and they might not understand how we solve our task individually, but you understand why we work in a task for the project. What is this task about? Okay, this task is meant to solve this particular issue so the team knows. Why the others work in something and understands? If we need to collaborate or not? Do we need to collaborate in this class or in this task or not? That's [00:28:21] a game changer [00:28:22] for me because if everyone understands they can be creative and propose better solutions, they can challenge the requirements. They can challenge everything.

Speaker1: [00:28:34] Will you allow your team members to spend like, maybe 30% of capacity to work on something that will improve their working efficiency instead of focusing on the specific user story?

Speaker2: [00:28:55] Yeah, I would be open to that. Maybe it's not the best or I don't know.

Speaker1: [00:28:59] Because you also mentioned that you ask your team members to propose two different solutions. Yeah, it will take time actually, but the output would be better.

Speaker2: [00:29:09] Of course. Sometimes you should balance because we are not pressured by our product owner. I prefer that they think about two different solutions. The only one? But just because we have this flexibility. So if we are not tight in schedule, let's do it. We don't lose anything, and we. We lower the risk of creating a solution that doesn't work.

Speaker1: [00:29:40] Yeah. Okay.

Speaker2: [00:29:41] And we can also document it and say, look, we have two solutions. We went for the first one. I don't know. I think this way of proposing, suggesting, challenging, only can happen if everyone understands what we are working on altogether. Everyone understands the project goals.

Speaker1: [00:30:12] But this project goal is for the high level.

Speaker2: [00:30:18] Epic, epic level.

Speaker2: [00:30:20] Epic level. Yeah. At epic level.

Speaker1: [00:30:22] Where do you have the specific one for the sprint. The sprint goal or?

Speaker2: [00:30:28] I would love to have that also. But we are not in a point that we can do that. I think ourtasks right now are part of with a common goal, but half of them without. So I would like to have all of them having this common goal for this sprint. We need to build this all together.

Speaker1: [00:31:04] Yeah, will you tell the team the goal every time? Or in which occasion you will emphasize on this goal because you really focus on this point. So how do you do that, that you can persuade or you can make sure the team members really align for this goal?

Speaker2: [00:31:32] Yeah. Sprint planning. I think it's really a good meeting wherewe explain why are we are doing a specific user story?

Speaker2: [00:31:49] Sprint Review I think it's going to give also very good results on that. Sprint closing worked very well so far because it's sprint planning and Sprint closing worked very well for for a common understanding.

Speaker1: [00:32:04] So every time you have this meeting you will tell the goal.

Speaker2: [00:32:07] Yeah I think that. And they are not longer meetings but 30 minutes only. So it doesn't matter. It's not much.

Speaker2: [00:32:16] I think it's. Yeah, it's. It's where we emphasize the. The goal for in spring planning for the next week. Having as a reference to the big goal. That is the big one. And every week the goal changes usually. So not the one for the epic, but the one for the week usually. Okay. We have seen that there are so many issues in this area. We we're going to focus on these other one or while they are being solved. I don't know.

Speaker1: [00:32:50] How do you respond to these changes?

Speaker2: [00:32:54] And you changed the priority of the user stories most of the times.

Speaker1: [00:33:01] But are you happy to do that?

Speaker2: [00:33:04] No.

Speaker1: [00:33:07] No, because you have to stop something.

Speaker2: [00:33:11] Yeah. Yeah. But at least this is sprint by sprint, not in the middle of the sprint.

Speaker2: [00:33:15] Yeah, yeah, yeah.

Speaker2: [00:33:17] So I'm not happy. But we have bloggers. We have we depend on the engineers in the production line to finish on documentation. Otherwise, we cannot continue. So we change the focus a bit maybe, or we are dependent on them.

Speaker1: [00:33:41] But that's the way Agile works. Because you break them down into the short iteration. So when you encounter the blockers, you can just stop and focus on the other things. But if you follow the waterfall, you're just stuck.

Speaker2: [00:34:00] Then it's working. It's working. If it's the purpose. Yeah. We are very agile on on changing the focus.

Speaker1: [00:34:09] Yes, exactly. Swifter.

Speaker2: [00:34:16] If all the team understands what is the goal, it's easier for them to change direction and prioritize for another goal than the other. Otherwise, if you don't understand and the goal changes, the sprint goal changes. The developers, for example. They don't they have not seen the the, the main goal. Oh, now we are changing completely. Why? No idea. Okay. I'm just going to deliver the new task. No, no. Challenge it. Yeah, we are changing direction. But if you feel like it's wrong, challenge it. And that can only happen if there is a common understanding.

Speaker1: [00:35:08] Yeah exactly. I agree with you. Okay, cool. I think we have everything.

Speaker2: [00:35:15] Yeah.

Agile interview with Participant Four (Speaker 1: Researcher; Speaker 2: Participant)

Speaker1: [00:00:00] Yes. Could you please tell me something about your projects that are running under the Agile framework?

Speaker2: [00:00:11] I'm in two projects and in the one we are trying to work Agile, we are trying to implement it. We started with we are in Sprint numbers two and three, and it's like an experiment for now. [00:00:32] So we are still adjusting. So [00:00:35] we are doing baby steps and the other one is one that we are together, which is I think much more mature, but at the same time, because now I moved from a big project, like ABC I'm in the BCD Yeah, because we don't have a scrum master. I will say it is not as structured as it was when I left the bird. Yeah, but we are working with Agile and Scrum and most of the scrum ceremonies. Yes.

Speaker1: [00:01:10] And how do you feel about this Agile or Scrum method?

Speaker2: [00:01:15] I mean, in general, I think there is a great value. It has a complexity in a way, and it sometimes adds a bit of a workload because you need, for instance, to spend time on refinement, especially in the one project that we are in the third sprint right now in the 24A. We don't have a scrum master. The product owner is not really there. So we are overextending ourselves, for instance, with D to make it work. So we spend time in refinement, we spend time in collecting, the backlog, and we are trying to do everything together with our workload. So it's a little bit adding like a hassle, let's say, like a burden, a weight on our shoulders. But at the same time, I think it's accelerated the workload, and there is a nice communication flow. At the same time, we are working on digital solutions, so we don't really we cannot see the future further than maybe three, or four weeks. So working Agile is actually helping you to break down and structure the short-term future. Yes. And you're already there even further without realizing it because it helps you make some stable baby steps.

Speaker1: [00:02:45] So when you try to implement these agile or scrum structures to the project, for example, with D. Which part do you feel is the most difficult part. Or any part that you think, would be challenging for you to make this kind of agile transformation?

Speaker2: [00:03:14] I think the number one obstacle is the fact that we are not 100% allocated. All of us. Some are 100%. And that's actually quite challenging because I didn't know that I heard that recently, that it's good to have as a principle, one enabler or user story, uh, active per time. So, of course, some of the enablers that are smaller, maybe there are two at the time, but if you add that

myself, for instance, I'm in two projects if that happens to my other projects. So suddenly, I have four enablers active at the time, and that's only because I'm working on two agile projects at the same time. So that's, for me, a bigger constraint. But if we isolate and imagine that we are 100% there, I think the tough thing is that when you don't have people getting the role, like you don't have a scrum master, we don't have a product owner. Maybe we do have a product owner, but he doesn't have the time, so we don't have the prioritization and guidance from them. So instead of relying on a designer working with data engineering and data scientists focusing there and performing there, we are trying to overstretch and help on these things. So I think that could depend on the situation for the biggest obstacles.

Speaker1: [00:04:54] Yeah. It seems that kind of time allocation for people to really take the value out of this scrum. If you are not fully allocated to the project, sometimes you will feel confused about the priority among the different projects.

Speaker2: [00:05:15] Yeah. Time allocation and the strictly defined role and people dedicated to their role.

Speaker1: [00:05:25] Okay, so how about the other team members? If they are 100% allocated to the project, will they have the same problem? Think they will be confused.

Speaker2: [00:05:41] Are you referring to the team members or.

Speaker1: [00:05:44] The team members. According to your observation, observation, or something.

Speaker2: [00:05:51] I think it's harder to talk regarding our Indian colleagues because it is much closer with them because they are working more in a technical aspect. But this is, for instance, already a piece of evidence that something is not there in the Agile setup because I should have also been closer, right? And in the Agile world, all of us are working together for a [00:06:15] common goal and. [00:06:18] So I cannot tell you about them. But for instance, adding V to the team now that he's also full-time, I think the three of us, V, D, and I are more aligned. So we are aligned on that matter. So I wouldn't say that could be an obstacle if we were if the others, like 100%, are located. Yeah.

Speaker1: [00:06:42] So according to so far, what you have learned from the agile activities like different meetings or practices, which one do you think is the most valuable or important for you?

Speaker2: [00:06:58] And are you referring to the ceremonies or?

Speaker1: [00:06:59] Anything, the activities that you can think?

Speaker2: [00:07:05] That's a good question.

Speaker2: [00:07:09] Actually, I always look forward to the sprint closure and sprint review.

Speaker2: [00:07:18] And the reason is, first of all. I'm getting a better picture also because I'm not technical, so I'm getting a slightly better picture of what is happening, what the status is, and how is the progress. Because when they are talking about our Indian colleagues and the developers are talking about the technical aspects of the status, I don't really understand what is happening, the check-in, but as soon as I see something complete, I get a better picture. So I really like the sprint closures and the sprint reviews because, in the reviews, we actually change them a bit. We don't do a demo yet because now we are experimenting, but we sort of like trying to capture the highlights and what we did in a presentation that we share with the product owner and so on, because they don't have the time to

attend those meetings. So I actually get some nice sprint screens, maybe from their code or like some diagrams, some architect diagrams. So it has a complete picture of that two weeks for what they did. And for me, it's much easier to understand. So those two elements of Agile are my favorite.

Speaker1: [00:08:43] So actually you can learn something through this process, right? Yeah. Okay. So have you ever taken some kind of meeting, like a retrospective?

Speaker2: [00:09:01] Yeah, I actually tried to do your sort of setup.

Speaker1: [00:09:07] Okay. What do you think about it?

Speaker2: [00:09:09] And we use the ADO board. At least I try to create, like, a template similar to yours. Uh, I actually really like it. And as we don't have a scrum master to be there as a facilitator, me and David, we are equally jumping into that. Yeah. The challenge of the retrospective is that we don't want to in a way force everybody to have some individual brainstorming on what went well and what didn't go well. So we are taking it as an open discussion. Uh, our Indian colleagues are less interactive in that session.

Speaker2: [00:09:50] And usually, it's me and David ending up reflecting. Because we try to shorten the meetings as much as we can. So we have sprint closing and retrospective in half an hour, which is very short, but that's because of the allocation. So if we really rush and then it's just me and David actually, for me it's also very insightful because we do some nice reflections, but at the same time I don't like that we don't try to hear what the others think, but at the same time that's also because we did it only twice. So I also left space-time for our Indian colleagues to understand how the setup could be and what they could talk about and give them the space to be able to say what to bring to the table. So our reflections, I didn't like this. It didn't work nicely because usually they bring positive things and I think that's also cultural.

Speaker1: [00:10:59] Some cultural differences. So I'm asking this question because I like to understand if you have ever implemented any activities or tried to figure out some solutions by yourself to try to Like behave like a self-organizing team. Because it's a very important value for agile that the team can choose whatever they want to work and then they have that kind of autonomy and then they can sell, they can do the self-learning, they can gradually develop themselves, the whole team efficiency. So I'm thinking about whether have you ever implemented or tried to figure out to make this kind of self-learning or self-improvement.

Speaker2: [00:12:10] Self-improvement is literally for me, right?

Speaker3: [00:12:15] Uh, yes. Or the whole team. Whatever you think. Maybe your team member is cooperating with each other. They can, for example, in the previous iteration, figure out that they don't have enough time to fix a bug or do something. And during the retrospective, you talk about this, and in the next iteration, you assign like 10% of your whole working time to fixing bugs or something like this, kind of improvements, self-learning, self-improvements.

Speaker2: [00:12:49] So self-learning, is part of bugs and it's not about training?

Speaker1: [00:12:53] It also includes some kind of learning from your past experiences.

Speaker2: [00:13:04] So it's sort of like reflection time.

Speaker1: [00:13:08] And also something regarding learning some new knowledge to improve your team's output.

Speaker2: [00:13:19] I will say because I cannot really think of something, I will say no and I will just say that things are going a little bit fast because you have two weeks in the beginning, you are basically in my UX world. I invest my time in understanding the tasks that I have and send the invites and prepare for the invites without even realizing it's the end. Yep. And then you're over and over and I don't really dedicate the time for improvements in the way of self-reflecting and self-learning together with the team as well.

Speaker2: [00:14:07] Maybe. But I don't know if that's we try to meet not regularly with D and have some discussions like now I just came up with a meeting that we had some sort of like frustration and I suggested like, Hey, what if we try this in the way we are working so we can take that? So sort of like improving this agile setup. But does it go under the umbrella? Does that answer your question?

Speaker1: [00:14:35] Yeah. Because you raise the issue that you don't have time to actually learn something to improve. Yeah, but if you have time or could you propose some solution to solve this issue or do you think that kind of self-learning is important for you? If you have time, how will you do that?

Speaker2: [00:15:04] Oh, that's a very good question. I think that's tough because it really varies. But right now, in the nature of the tasks that I have, I would have maybe liked to have even fewer tasks so I could actually invest that remaining time in the curiosity of exploring, listening, what is happening, capturing some of the hurdles, and maybe investigating in a casual way how we could maybe overcome because now everything is a little bit pressured. Now that we came with frustration, we had another meeting, so we discussed a little bit about that. I suggested something he's going to try to see if that can work and then maybe we solve it. Maybe not. But that's not like a systematic way of tackling. So maybe I would have liked to digest that information, have some reflection time, you know, sometimes that I'm just sitting and thinking.

Speaker1: [00:16:20] Yeah.

Speaker2: [00:16:21] This time you find me outside of work. But I would like to use my brain power. I try to work for thinking regarding work, so maybe making sure to leave even more time. It's very hard to say When is this 80% allocation to leave this 20% for backup spillovers, supporting your colleague's consultation and stuff. So I sense that maybe I'm closer to 100% allocated to both of my projects, but at the same time, I don't it's not very tangible. You cannot really maybe reason the border and I cannot really read those graphs, but I think maybe if I really had this 20%, maybe I would have had the time to reflect on it and spend some time with my colleagues grabbing a coffee and thinking, Uh, yeah, that happened like, you know, like knowledge sharing, sort of.

Speaker1: [00:17:22] And yeah. So you like this kind of learning? Yeah. Or we say the backup time for developing and self-learning some kind.

Speaker2: [00:17:35] I like to learn from talking with people. So that's my preferable, preferable way of learning. Yeah. Saving time and having some discussions.

Speaker1: [00:17:49] Yeah.

Speaker2: [00:17:50] Also, I brought some learnings so I might changes. Like something cool. Are we doing the X project? I am. That's sort of like this self-reflection and like whenever I hear something or I find something interesting, I say, Hey, maybe we could do that. We do. That works nicely. What about trying? And we are experimenting. So that's also a way to both ways circulate information, right? But usually, you are 100% allocated. So then, um, you don't have this privilege of exchanging, Uh, okay. That worked. Nice learnings, let's say from project to project, uh, in a simultaneous way. Yeah.

Speaker1: [00:18:35] Yeah. And yeah, that's also an interesting thing or topic to mention for the very traditional scrum method we just deliver. I think after each iteration and there is no stop actually. But when we implement the SAFe. That's another structure or methodology. They actually have what we say, the IP, Innovation, and planning, section. So during this iteration, you can really do something. You like to try to improve. So that would be another kind of solution to solve this Kind of self-learning issue. But we will see. I'm not very sure if it really works.

Speaker1: [00:19:29] Maybe the spillover from the previous iterations will take up most of this IP. Yeah. So we will see that. But that's a good point to think about like actually the team as a self-organizing or self-learning team, they really need time to do some reflection to like sharpen the tool to improve the working efficiency in the future.

Speaker1: [00:19:57] Yes. I agree with you.

Speaker1: [00:20:00] And the next question would be regarding responding to the changes. Yes. So I mean, have you ever received the requirements changes from the product owner or customers? During the iteration?

Speaker1: [00:20:19] Or will you have this kind of issue with too much frequent changes? And how do you how did you react to these kinds of changes? Yeah.

Speaker2: [00:20:33] I mean there is two sorts of directions of change. How I see it. One is like a complete different change, like, okay, let's add some more stuff. That's a change in the plan. And my approach of those kinds of changes that are coming a little bit out of the blue, I'm actually really trying to just push them away and sort of like clean the way we should always keep in our back, in our head in the project that we are talking about. We do have a product owner, but he's not actively there, so we will never receive some from a product owner some direction. So it's coming from also random directions, this kind of input.

Speaker2: [00:21:23] I'm always trying to put like a blocker there and just say okay, cool, we will put it in the backlog. That's my reply always. Now there is also the other nature of change which is about investigating something, and I learn something. So that's sort of like that's why we work agile. And then I'm of course much open to those changes. Uh, that could be I learn something, I circulate it with D. Because he has the technical knowledge. He thought of something and then he pushed it back like, Oh, what if we also need to do that? So then I'm actually assessing if that's a natural continuation of this specific enabler or if it's something like out of the blue, something brand new. Because if it's something brand new, I actually put it in the backlog, but it's something that I keep in mind to push in the next sprint, right?

Speaker1: [00:22:26] Yeah.

Speaker2: [00:22:27] But if it's something that actually it's ad hoc and it does make sense, I will immediately, sort of embrace it with what I'm doing at that moment. Uh, it's a little bit different. I will say the UX nature of work versus developer work because you cannot really find bugs, that is a priority. So you have to clean. It's about you exploring and you learning and you are wiser and smarter, so then you take some actions based on that. So, it's a little bit easier to handle those changes, but definitely, I don't have people shooting. Uh oh, you do this or this. What about this? This definitely. I don't have this. Uh, unstructured way of communication. And if I do have it, it's like we will put it in the backlog. Uh, yeah, so that's, that's how change wise my work looks like.

Speaker1: [00:23:29] Because of one of the agile manifesto is that we focus on responding to changes rather than following the plan. So what do you think about the value you can get from this kind of responding to changes?

Speaker2: [00:23:51] I think it makes very good sense and that's why I split the changes in two paths like an embrace, like the manifesto of responding to changes. That's sort of like we scratch the surface of something and then that unfolds a new world and we need to embrace it.

Speaker1: [00:24:10] Yeah.

Speaker2: [00:24:11] And that's most likely is going to be the one that you are adding things in the following sprints. And that's 90% of how I work. But there is also like this 10% of people just having some cool ideas out of the blue or what if we do? This is so cool, which is a cool idea, but it's like, okay, but without investigation right now. And you know, it's like about looking at thin air, then you just park it on the side. Because for me, that's not the same density of information.

Speaker2: [00:24:46] I will say I'm 80, 90% to the manifesto side, but I'm also like filtering some stuff. It's like I'm not like if a person comes and say, okay, out of the blue, let's create a machine that does everything tomorrow. That's what we are trying to do. So that's cool. Let's park it and let's keep moving because that's our goal. So I'm not like. I also feel a little bit maybe that's not good now to think about for innovation.

Speaker1: [00:25:15] But that's really good. I think so I think would be my last question. It's almost half hour. So the last question would be in future if you want to continue working under this agile structure, what kind of activities or value you think you should keep on doing?

Speaker2: [00:25:52] And again, activities. Do we mean ceremony? So everything. Everything okay. I don't know if it's an activity, but I will definitely say that we need to have a defined product owner and scrum master because they need to. So the way instead of just letting the teams. So that's something that I would like to see forward. Um, and I would like to see as well. That's a lot of stuff. I'm trying to prioritize it in my head.

Speaker1: [00:26:30] Yeah, you can just list the highest priority.

Speaker2: [00:26:34] I mean that's the top thing that is coming from me. Product owner, Scrum Master a very important. And they need to be there. Yeah. And then also like this 80, 20% allocation, that's something that I think I need training because I don't it's intangibles I don't understand. So that's something that I would like to see. I believe if a Scrum master and a product owner had experienced this

80 over 20, they could say raise the red flag. So that could be linked with the first thing, but at least 80 over 20. I would like to see it more.

Speaker1: [00:27:10] Yeah. So 80% means you need some time to do some self-learning or reflection. Yeah.

Speaker2: [00:27:19] 20% is not really allocated, but it's not because you are sitting drinking coffee in front of the computer. It's because these time is spent on exchanging. And as a colleague of yours had an issue. So it's just asking you instead of rushing and being.

Speaker1: [00:27:35] Yeah, sure.

Speaker2: [00:27:36] At the same time, flipping a little bit, the coin I would like, I know that spillovers are bad, but actually I like the flexibility of something, let's be honest. It wasn't success. Instead of rushing it to make sure to finish it for the sake of finishing it at the end of two weeks. I like that we can spill it over to the next sprint. I know that it's bad, but it's sort of like you don't do this two weeks too. It's not a harsh deadline that you need to have stress. So I like somehow to do a good robust plan knowing that there will be spillover.

Speaker1: [00:28:21] Um, yeah.

Speaker2: [00:28:22] Yeah. Usually you don't want an agile spillover. But it's like I know that if you have this 20%, maybe you will use this 20% for spillover. Right? But this spillover for me, it needs to always be no, this 20. Sorry. No, it's very what I'm saying. 20% is about could be spillover, could be talking, consulting with colleagues, digesting information. But I wouldn't like that to be spillovers because then it's actually ending up being 100% into the project. So it's not really solving the problem. So I would like to see more spillover in a way that is not a harsh deadline for me. It's also I'm exploring. So sometimes there's more depth and we need to dive into, sometimes not. So it's very tough to assess this thing. Maybe it is with the right tools and Scrum master knows. But yeah, that's the information that I have for now.

Speaker1: [00:29:21] Regarding these kinds of workloads, when you do your sprint planning meeting, who will assign the task to you? Will you get the task by yourself or.

Speaker2: [00:29:35] Yes.

Speaker1: [00:29:36] Okay.

Speaker1: [00:29:37] So maybe in the first iteration you think you can complete eight. But by the end of that iteration you review the whole iteration. You actually completed six tasks. So maybe in the next iteration, when you do this sprint planning meeting, you think, Oh yeah, previously we, we planned eight, but we completed only six and maybe this sprint, we just take six instead of eight. So yeah, there will be some kind of self learning process.

Speaker2: [00:30:11] Exactly. Yeah.

Speaker2: [00:30:12] So but I think, I think the scrum master is the same outside work as well. You always think, oh yeah, I can handle it and you just add more and more stuff on your shoulders. You it's nice to have someone else doing that for you because you, you're like, Oh yeah, I can handle it. Just give it, give it. Yeah. Because I don't have this self-learning. Now we're on the third sprint.

Speaker1: [00:30:34] Yes.

Speaker2: [00:30:35] And I keep doing the same. I'm aware that I shouldn't do that, but I'm like. Oh yeah, it's fine, I can handle it. But if someone else tells you, then you're at the moment that you realize, okay, we stop now, but now there is not really someone telling me, Don't do that.

Speaker2: [00:30:55] So that's why I think having these roles play an important element.

Speaker1: [00:31:03] Yes. Okay.

Speaker2: [00:31:04] It's, it's exactly what you said that what we should do.

Speaker2: [00:31:08] Yeah. But it's like sometimes, it's the human nature.

Speaker1: [00:31:13] Yeah. Some kind of, What do we say? The consistency or consequence. You just follow the previous schedule or thinking. Would be the important part of the Agile, I think. Self learning retrospective reflection.

Speaker2: [00:31:38] Yeah, that would be important.

Speaker1: [00:31:40] Okay, cool. Thank you so much for your time.

Speaker2: [00:31:43] You're welcome. I hope you got your answers actually.

Agile interview with Participant Five (Speaker 1: Researcher; Speaker 2: Participant)

Speaker1: [00:00:00] Okay, great. The first question would be. What do you think about the Agile values?

Speaker2: [00:00:09] Uh, that's a very open question. Um. Which ones in particular? Let me ask that.

Speaker1: [00:00:17] Maybe you can just talk about responding to changes over following a plan.

Speaker2: [00:00:22] Okay. Yes. So. I appreciate the value of running Agile projects in that delivery plan. I won't say it becomes more flexible, but you have a better guarantee of delivering the right product on the first go if you execute an Agile project in the right way. That's one point.

Speaker1: [00:01:01] And regarding this kind of delivery activities or delivery work, can you elaborate on what kind of detailed activities or do you think that is important?

Speaker2: [00:01:21] Well, in an agile delivery model, I think it's very important that a proper, um, preparation of a backlog is the key to being able to deliver some real value. Because most agile projects are focused on the scrum methodology and the methodology of delivering value in increments and moving results forward on a regular basis, which are good principles. But you need to know how to move forward. Otherwise, you're just moving something forward for the purpose of moving it, not for the purpose of actually realizing its value. And that's at least one of the pitfalls I find from many agile projects they fail to identify and prepare their backlog properly before they start delivering. So they deliver towards an intended value, but not the expected value.

Speaker1: [00:02:30] Yeah. Okay.

Speaker1: [00:02:32] And regarding these kinds of values, have you ever encountered any issues when you want to deliver some value at the very beginning? But finally, you find that you have to change accordingly to the input of the customers.

Speaker2: [00:02:54] I've never found a problem changing when it's the customer who changes their expectations or requirements. But oftentimes, I feel that projects may run into complications because the project thinks that it's the customer that changed the input. But maybe it's just the project that didn't understand the customer's requirements in the first place. So it's sort of a complicated statement that I'm giving you here, but it's because I believe that. [00:03:32]There's a big difference between the delivery team not understanding the requirements sufficiently and then delivering something wrong. [00:03:39] Going back to the customer saying, Oh, we have to change because you're getting something else. Then the customer is saying; Yesterday I wanted a green button, Today I want a red button. It's a clear example of just the change in requirement, and then it's okay that you're able to change the plan.

Speaker1: [00:04:00] Okay. If this kind of change occurs in a project, does it have any positive or negative effect on your Agile team?

Speaker2: [00:04:19] That's a really good question. I don't think you can say that it hasn't a positive or negative effect on the team because changes will always incur some sort of negative effect. Because everyone hates changes, right? Especially if you're a delivery team. You delivered something, and then someone asked you to change it. That in itself is going to be negative. But I think if you work in a well-structured, agile process where there is a good and structured way of managing changes, then it will be less of a negative impact on the team because the team knows that they are just receiving a new requirement and that requirement then goes through the process of being designed, developed and tested and delivered. So it's really just a matter of, okay, what's on my Agile board this week? They just deliver it, and it's not really a problem. But if that process is not well executed or defined, then you end up having a project where changes become troublesome for the team because they receive a lot of changes, but they're not always in control of what they need to do to deliver them.

Speaker1: [00:05:53] Yeah. So they're kind of you mentioned the change management is something actually an independent part from that, the daily Agile or scrum process, so that the development team will not be influenced by these kinds of changes until they have the next like backlog refinement or spring planning meeting. So they know they got some new input or new task instead of some changes.

Speaker2: [00:06:26] Yeah, exactly. It's really a matter of making sure that the continuous loop of developing agile requires you to make sure that you have a good process for when we accept a new change or development item. Right. And if the change is well described and the process of managing the change or delivering the new feature or updating the feature is in control, then it's smooth sailing, and you just on a regular basis in your agile iterations or what you call them sprints or something like that. Then you can just take on new requirements or changes on a regular basis.

Speaker1: [00:07:16] Okay. So what do you think about the next question about the self-organizing teams or the Agile teams? Is it different from the team from the traditional project?

Speaker2: [00:07:37] Yes, I think so. Self-organizing teams are very crucial in delivering a successful output. But I also think self-organizing teams shouldn't only be limited to the Agile delivery model, so from a theoretical perspective, in a traditional project model where you deliver according to a plan in a

waterfall. I think a lot of time is wasted because the team isn't self-managing or self-driving. The delivery of their work. So if the team is actually able to deliver the work completed faster. Then the project doesn't gain the benefit of it if you're in the traditional model. Because you expect something in three months, so if you get it in one month, you're not going to. You won't be able to get the value out of it until you reach three months. You get the value out of it in the Agile model because you expect to deliver something. You think it will take three months, [00:08:51] but you're finished in one, and because that process is shorter, then you gain benefit from it right away. [00:08:57] And the self-managing teams are able to define better when and how they're going to deliver, whereas they are, they tend to, at least in my experience, only deliver when they are asked to if you follow the traditional plan because they have to be told when to deliver.

Speaker1: [00:09:21] And do you think currently or what kind of characteristics you think you can use to describe a self-organizing or agile team?

Speaker2: [00:09:36] They are driven by the completion of team-oriented goals. And they are self-motivated as individuals but also within the team.

Speaker2: [00:09:51] Yes, there is some kind of information alignment. They should have. Right?

Speaker2: [00:09:58] Yes. And it can both be based on their ways of working, so they all follow and understand the agile methods and ways they are working within their project, but they also understand each other, and they are also aware of the [00:10:12] different skills in the team [00:10:14] so that they can pass on work when they need the input from someone else.

Speaker1: [00:10:20] Okay.

Speaker2: [00:10:23] Regarding this kind of self-organizing team, to what extent do you think that they should have that kind of autonomy to manage themselves?

Speaker2: [00:10:36] And well, this is more of a personal view. I think they should be able to manage themselves almost completely. Okay. But it's also because they are only managing themselves within the confines of what happens within a sprint, for example, because the planning of the backlog and description of the requirements is not the team that delivers that. And that's where I see most of the sort of official value. You define what you need to get. How are you going to get value out of the project? That's the requirements and everything that happened before. But what happens in the infinite loop where they just deliver the solutions that are best done if it's just managed by the team? And they can be left by themselves. [00:11:28] just to deliver. But they need to have clear goals to deliver on. [00:11:32]

Speaker1: [00:11:32] So the goal would be the most important part of self-organizing. And one more question. And what do you think about the relationship between what we discussed previously, the responding to changes and self-organizing teams?

Speaker2: [00:12:01] Well, I think the self-organizing teams will be better at managing changes because they are used to having the responsibility of managing the change themselves, whereas a traditional team will be given a change. And that change might not work well within where they are in the process. So think again about someone that needs to deliver something in three months, and we now introduce a change. That means that an entire month's worth of work is lost, and they had to start over. [00:12:46]

Whereas in the self-organizing team, they might be able actually to modify how this change is being introduced, so they lose less work, [00:12:54] right? Because they're not focused on delivering something in three months and planning their work accordingly. They're focused on just delivering that value within whatever time it takes them.

Speaker1: [00:13:07] Yes. Agree.

Speaker1: [00:13:10] Do you have any idea about or what you think about that kind of self-learning in the organization? Because every time when you have that kind of retrospective meeting, your whole team do this kind of retrospective, and they will make some improvement. So what do you think about these kinds of retrospectives? Do you think it's valuable for the team to respond to changes, to learn by themselves to develop by themselves?

Speaker2: [00:13:47] I think it's very valuable, and it's one of the really key aspects of getting value out of an Agile project is that the Agile project, if it follows certain sort of theory and models, actually relies on this feedback loop that you continuously are reviewing and improving your ways of working. This is not very common in traditional project execution because the feedback you get is often when the problem is already over and too late to manage, and you might only realize that you need to change something when you're already past the deadline. Whereas here, you build it into the natural delivery in the Agile model, making it just a natural part of how you progress that you continuously improve and make sure that everyone gets better and moves faster and is able to deliver onto the common goals.

Speaker1: [00:14:54] Yeah. And according to your past experiences, what kind of self-learning would be? You can say the most valuable one, or you will keep on doing it in the future when you have other new projects.

Speaker2: [00:15:12] I think I've always embraced agile ways of delivering. Even before I started working on Agile projects. And the reason for it is that it allows for change. I don't like to embrace change, but I like that you allow for it, whereas it's usually more strict in traditional project execution where you have sort of it's a long process. To present an issue, discuss what the change impact is going to be, and then actually finally get allowed to implement it. Whereas in the Agile project delivery model, your customers and project members are so aware that changes can occur that you rely more on the execution method rather than the actual product you're delivering. Therefore, the changes are just easily taken into the project. And yeah, and I like that. That's the real value I find, okay, that the customer is able to continuously evaluate if they're getting the right output, thus modifying the way they want it so that when the final delivery is then made, they get the right output in the first run.

Speaker1: [00:16:41] Okay. Yeah. That's great. And also for this kind of self-learning, of course, you have if you manage the project and your team members want to develop themselves to learn something by themselves, it will cost money, the time. So what do you think about balancing this kind of self-learning time and the delivery time? Like if you have a kind of pressure that you have to deliver something. So how can you balance them?

Speaker2: [00:17:21] Well, my experience tells me that you should always make sure that you work with improvement along the way because otherwise, you're just going to keep introducing the loss of time in the future as well. My point is that if there's something that you're doing that can be done smarter, or you know what I'm doing now, this way, it takes very long. It's super manual. It takes me at least five

days. But if you don't work to improve that, then it's going to take five days continuously. So it's going to every time we need to do that, prolong the delivery time of our project. Whereas if you spend a little time figuring out how to optimize it, then you're going to be able to deliver more and faster in the future.

Speaker1: [00:18:10] Okay, So maybe that's also important to explain this kind of self-learning to the customer because the customer will pay the money for this.

Speaker2: [00:18:25] Yes, but it's always a balance, right? Because I don't believe that the customer needs to pay for a lot of self-learning. Yeah, but it's important. So it's not like it's 50% of the time spent is on just learning and educating. Yeah, it's more a matter of saying. Last month we delivered 100 hours. We had to use ten of these hours that were actually done on something else. So I would never say it doesn't really take up much more than 10% ever of the time spent. Yeah. And not even 10% most of the time.

Speaker1: [00:19:03] Yes. Okay.

Speaker1: [00:19:05] Okay. The last question. When you want to implement the Agile value from scratch, which the team has not implemented yet before, do you think educating your customers or product owner about this kind of agile value, responding to changes, and self-learning from the very beginning is important? And what would you do to persuade them to be transformed?

Speaker2: [00:19:44] Well, that's an ongoing discussion with the customers and stakeholders because, first of all, when you engage in an Agile delivery model on your project, it is something that the customer is somehow already agreeing to. Yeah. So it's more the case that you have to continually reassure them that they are actually getting value out of the delivery model. And I think the most important part is making sure that the value that they see is the actual product and not just the status reporting out of your Scrum board or Kanban board that things are now all in the done column. Because that's the common misconception that the Agile project delivers value by continuously ensuring that we deliver an outcome and not only output, but we also have to make sure that we can show the customer an actual tangible product that they have in their hands. So I would say that they can get on board. Not easily. They will still be reluctant to work with Agile because many of them don't understand why or how because it's still rather new, at least in our business and industry. But they need constant reassurance of the value that it delivers.

Speaker1: [00:21:18] Yeah. Yeah, I agree with you. That's very important. Okay, great.

Speaker1: [00:21:23] I think that's all the question.

Speaker1: [00:21:25] Thank you for your valuable input. You're welcome. Thank you. Great.

Agile interview with Participant Six (Speaker 1: Participant; Speaker 2: Researcher)

Speaker1: [00:00:02] Can see that I have. It's pretty new to this methodology. But yes, it's been a while in a lot of industries, but there is a change in the there's a shift in the thinking that earlier people were working in a waterfall model approach. But with the recent changes, of course, it gives a lot of flexibility. I can say that you have a very high-level overview of what people are doing. I can see that people are

more productive and they're very controlled. You know what your colleagues are doing, and you always feel that you're connected with this approach. In the previous processes, you had to wait till something you were waiting for some output. But here, this is not the case. You do the work, and then you fix it then and there. And that iteration concept, I really liked it.

Speaker2: [00:00:53] Okay. Regarding agile values, we always say that we need to respond to changes rather than follow the plan. In your opinion, what are the responding to changes in the Agile projects?

Speaker1: [00:01:18] We see changes, of course, like if you look at the change management in terms of it, that's pretty vast. You have to go through a lot of things. But when I see when it comes to agile, it gives a lot of flexibility. It gives a lot of openness, and it gives a lot of the same thing, the control panel. You have a lot of reviews, and the way people interact through that portal where you have this pull and push that DevOps ideology makes things much simpler so that people can be more attentive and can focus more on their project deliverables. So you don't need to have a dedicated person for the change. It just needs to have somebody who can drive and also, at the same point in time, make everyone aware that this is where the organization is heading. So it's not working in silos, which I can see it was earlier in the change management and but now it's more of a very open where everybody can see that, okay, if we follow the Azure DevOps or Agile methodology in for the change, everybody is aware of where we are leading.

Speaker2: [00:02:31] So according to what you said, it seems that you have ever worked on a Waterfall project and also on an Agile project. So you compare two kinds of change management, and you are more satisfied with an agile project with these kinds of changes because you have the flexibility, right?

Speaker1: [00:03:01] It's business actually means the entire project in the world, meaning wherever we see this as a success. Yes. So it adds a lot of value.

Speaker2: [00:03:10] Okay. If these kinds of changes occur in your project, does it affect your team positively or negatively whenever you think you figure out you have encountered a change?

Speaker1: [00:03:27] See any changes without prior notification will have some resistance. That's normal, irrespective of anything in the world. Okay. So first, if somebody has been working in one way for many, many years, the first question is why it is needed. So now the responsibility of the owner or maybe the company or whoever is the stakeholders should be aware of why we are changing it and why it is necessary. So, of course, it is very important. And I think that if people are aware, it is very easy to implement. If we just simply go ahead and just launch something that, okay, this is a project, and we are going to do it through Agile methodology, I don't think it's going to work. It will start, but eventually, we will have a lot of confusion. So it's very important to have that understanding in advance. That is what is agile, and we are comparing it with what and why we are doing it and where we will be heading if we follow the structure and onboard the people. I think it would be easy.

Speaker2: [00:04:33] Okay. This is a kind of process change. But regarding the requirement changes, have you ever encountered them during work, and how do you feel about them? Like your product owner just changed something during the iteration.

Speaker1: [00:04:58] Yeah. It all depends. On the type of project that we're doing. If we talk about a software development project where everybody works on bits and pieces parallelly. Then it might be very easy to change something for the product owner because everything can be decoupled into

individual pieces. But when we work on something which has a [00:05:21] dependency on something, [00:05:23] then at that point, if something changes by the product owners or by the reviewer or by the design specification owners, then it will have an impact. Then, in that case, the person who is actually leading the project or who is actually writing this design specification should be very much clear about what we are doing and why we are doing it. And if we make any kind of changes where it will, there will be an impact.

Speaker2: [00:05:53] Okay. You mentioned that we need to understand the background of these changes. In your opinion, do you think that this kind of agile method or methodology can help you to understand background knowledge of the changes?

Speaker1: [00:06:17] Can you repeat it one more time?

Speaker2: [00:06:20] Yeah. I mean, for these kinds of changes, you want to know the background of the knowledge, but comparing the traditional waterfall methodology with the Agile methodology, which methodology that you think will help you to understand the background of changes and how.

Speaker1: [00:06:42] When we're talking about understanding, I think both Agile and Waterfall are in the same place because it's only a matter of getting the requirement. What actually makes the difference is deliverables. When we do it in the waterfall, there is a lot of waste. When we do it in Agile, there are a lot of sprints, and there are a lot of controlled ways. Now, coming to your question regarding the requirements, I think it's a top-down approach. So whenever we start any project, the requirements have to be very specific. Of course, it should not be sealed that, yes, this is the path we are going to take. If there is any change, it should not be more than 10%. If we follow this approach, then it would be really nice. And that doesn't matter whether it's in waterfall or Scrum.

Speaker2: [00:07:27] Okay. Do you think that if we split these kinds of projects into many different iterations in this kind of way will contribute more flexibility to help the team to manage these kinds of changes because the iteration is very short? And by the end of the iteration, you can review and you can get feedback. 0r do you think that you just follow specifically the design specification from the very beginning and you develop, and with a long period of time, and in the end, you compare it, and you test it based on the design specification?

Speaker1: [00:08:15] Well, I can say both when it comes to requirements. It has to be in advance. And when it comes to developing something, it can be in a lot of iterations because there's something we need to do pre-checks, we need to test, then there will be bugs, we need to rework on those, and then we need to go back to the design specification again and see whether it's matching with it or not. So I can say the roof, the top should be the design specification or the requirement which has to be intact. And as I say, that we can only accept a change of 10%, maybe not more than that. If we are making a requirement that might change by 80% after five iterations, then it's not good. So we should follow some iteration of some requirement, which at least 10 to 15% changes is acceptable in this kind of project. And then when we move to the development, where we have a lot of sprints, of course, it makes it a lot of easy to discuss because now everybody is on the common page where everybody has read the requirements.

Speaker2: [00:09:12] Okay, great. And for this kind of change, we need to figure out why does this change happen and which kind of activities do you think will help us to identify these changes during the whole Agile or during the iteration?

Speaker1: [00:09:39] I think in XXX we have a lot of projects that we have done. If we go ahead and get the learnings from there and just take inspiration from any one of these sites on how it has been implemented and just try to match our requirement closely with that, then it would be it would help.

Speaker2: [00:09:56] Okay. Regarding the internal team members, if you figure out the changes that happened during the iteration, how will you communicate with each other and figure out the solution?

Speaker1: [00:10:19] You know, for me, I will call it a workshop or maybe a virtual or in-person meeting where we are going to spend half a day discussing that. This is what we had, and this is what will be. And then we map the changes, and we take an agreement that, yes, everybody understood that because of this change, who needs to do what? Then we go back to our desks and then implement the changes. And again, after iteration, when we come back, the output should match the change requirement. So that's what I will do if I have to run it.

Speaker2: [00:10:54] Okay. Do you think that kind of daily stand-up will be useful to align the information together and manage the changes?

Speaker1: [00:11:03] I can say it's a heartbeat of this Agile methodology, I can say. Because it always had connectedness, especially when these days we are working in hybrid. It has that connection with all the project teams, irrespective of the location. And that is very much useful. Sometime we might not have the proper updates. Maybe we are stuck in some areas where things are not moving, but that's fine. That openness and just to raise a hand and say that this is where things are going wrong, and this is where things are. We need help or things are absolutely going fine. That makes a lot of sense. A lot of clarity

Speaker2: [00:11:41] Yeah. And also, regarding the team, do you think that the team, especially the self-organizing teams, plays an important role in this kind of agile methodology?

Speaker1: [00:12:05] Okay. Yeah. Because when we talk about Agile, there are a lot of we need to know what is the Epics and how things are interrelated with each other. So a self-motivated and independent team is required.

Speaker2: [00:12:17] Yeah. Okay. So what kind of things that you think are important to help the team to understand they are self-organizing? They have the autonomy to do whatever they think they can do during the iteration. So what kind of thing do you think is important for the team members to understand?

Speaker1: [00:12:49] First of all, I feel that if somebody is pretty new to this ideology, they need to be trained. And when I'm saying training, it's not only the tool. It's the ideology and the theory. If I have to take a training, then I will give an example of a live project that has been done like this, and this is the time frame that we took. So, you know, you need to tell a story that motivates everyone. And then we need also to break down our existing projects, tasks, or activities so that people can pull things by themselves rather than waiting. They pull things from the board and then develop them. So two things. One is understanding the process in terms of the training and the hands-on it's much needed. And, of

course, the big picture, the vision that where we are trying to do in terms of the project quality and in terms of cost saving to the company as well.

Speaker2: [00:13:43] Okay. And do you think that this kind of self-organizing thing or the autonomy that you have under this kind of agile methodology is a valuable thing for you to manage your daily work like you? If you have autonomy, you can do it better.

Speaker1: [00:14:16] Yeah, go ahead. What was that?

Speaker2: [00:14:17] Yeah. So. Well, I mean, I like to know how you think you can implement this kind of autonomy in your team, in your daily work.

Speaker1: [00:14:31] Okay. For example, if I don't have project experience at that point of time, I'll, of course, take an example. I mean, something always goes in my mind that if I give a story that a company has made their website within six months with all these features, with five resources with this amount of money. And with hundreds of epics and 200 tasks and with the Agile method with 50 iterations. But what does that mean? Then we tell a story. We do a lot of workshops that let's do a dummy Agile. So eight hours in this room. This is a project duration. We'll have ten people, and two people each need to deliver one iteration. You have sticky notes, go ahead and stick what you want. Then other people challenge one. People will have a requirement. Specialist one person will be the reviewer. So like this, I'll try to create an environment where people will understand that this is where we are heading. And then, we look at the board when it comes to the tool. And then again, another workshop on how we implement this ideology in our existing project? So like that, if I have to take a call, I would take it in this direction.

Speaker2: [00:15:58] Okay, great. And how do you think that self-learning in your work? How much time will you spend on self-learning during the iteration that you think will be good or enough for you to develop yourself?

Speaker1: [00:16:23] Now in for learning. There are two aspects. [00:16:29] One is technical. [00:16:30] When we are making something, there is a lot of time spent on the internet or books. How do we solve it? [00:16:36] And the second is learning the process. [00:16:38] Yes. So again, if the people are trained, it would be much easy, or else they go to their colleagues and ask how. How do I link this epic to the enabler? Or how do I link this test plan to something? So. By doing things again and again. Again. One person will have an idea of that. This is how things are going to be.

Speaker1: [00:16:59] So yeah, you will always get self-learning by doing things. Whenever you do, put some of your hands onto something, and if you do it 100 times, you will always get an idea that this is how it works. But it's not a replacement for training because I always give an emphasis on training, providing training to the tools, to the methodology, and of course, to the right people. If somebody wants to be a doctor and if you tell them to be an engineer, that doesn't work, something like that. So yes, self-learning is always there. When we go through this agile methodology, we always have something to deliver because that's the time frame within a day or maybe within two weeks, depending on the iteration. So people will have that notion in the back of their minds. We need to deliver, and let's go ahead and just not sit instead. If we don't have any idea about something, go ahead and ask someone who knows it.

Speaker2: [00:18:00] Yeah, because just as what you did in what you are doing in the current iteration, like you created something like an improvement work item, that's actually this kind of behavior or activity is actually built on that kind of self-organizing mind or self-learning mind. So yeah, I think that's a very good thing. Could you please give me more ideas about when you created the enabler or executed this kind of activity? What kind of thing popped up in your mind? Like, why do you think you need to do this? And yeah, why do you think that's necessary to create a new or improvement work item here?

Speaker1: [00:18:59] The finer you make the end product, it will be beneficial for the company. So it's always about making the final product in a much more refined way. So the way I thought is that okay when I'm done with my deliverables, there is something which can be done in the later part. So instead of that, let's focus on something which can add value. So I used to think from that angle, and then when something came up in my mind, which thing can be a part of the board? I used to create that as an improvement item. It can be in updating the scripts, updating the comments or writing SOPs or creating test plans, etcetera. [00:19:41] So you will only get that idea once you know where the project is heading, what the end goal we are trying to achieve, and what is at stake. [00:19:50]

Speaker2: [00:19:51] Yes, exactly.

Speaker1: [00:19:53] Things will not come when somebody doesn't have that understanding. If somebody is like, I need to do ABC, done? No, but there are more. You can do A.1, you can do B.2. So you can also do the subparts. So once somebody has a clear picture, they can think more and come up with this improvement item. Actually, that's the approach I took.

Speaker2: [00:20:15] Yeah, I totally agree with you because that kind of goal of the project is very important. Team members can follow this kind of goal to think about what kind of work they need to do and they can prioritize based on this kind of goal instead of asking the product owner. That's really important to understand this goal from the very beginning. Yeah, I totally agree with you on this part. Okay. And the next question would be what kind of things you think are valuable and what you have learned through past experiences, the best practices, and what you want to keep on doing in the future.

Speaker1: [00:21:17] First of all, what mistakes or errors we have made? I mean, let's try not to repeat it again. That's what. Because if it's a new experience we have gained, so we should learn from it and be a better version of yesterday. That's one.

Speaker1: [00:21:35] Second. I felt it was very important. Like we do a lot of discussion in this Agile method, but at some point in time, we need to take a call that we go in this direction. That is sometimes I believe some projects lack and that's where people get started confused.

Speaker1: [00:21:58] So if I will work on a different project, which is on Agile methodology going forward, then I'll, of course, put stress on creating the platform in the beginning because Agile is a kind of wheel that keeps on going. It can be fast, it can be slow, but once you have the platform ready when I'm saying platform in terms of requirements, resources, clarity, tools, test environments, servers, everything. Then it's just a matter of motivating, motivating the people and just keeping the ball rolling. And if you have somebody on the project who knows what is good or not so good. Has he, has she experienced in the previous project? Then that person can act as a leader to say I think that in this direction, something might go wrong because I have experienced the same. That's how that person can also mentor there. I won't say [00:23:03] mentor can [00:23:06] encourage the rest of the project team

members that we should follow this. Of course, we can always welcome all the questions and challenges, but in the end, it's very important to stick to one solution because meeting takes a lot of energy and effort. And after that, if we do not have some strict actions to follow, then it will not work. So follow actions, and deadlines and just learn from the mistakes.

Speaker2: [00:23:32] Okay. You also mentioned that at the very beginning having a platform to get things ready is important. But do you think that's a kind of anti-pattern for the agile value, like responding to changes? Because, based on that, you can make this different iteration, you have some kind of output improvement, and then you go back to implement them into the next iteration?

Speaker1: [00:24:03] Yeah. So we'll try to make the requirement close to 80%. That's what I said. Like I'll always give a 10% or 10% buffer, but we'll try to do the improvement items within this 10%. So then we can always go back and work on our improvement. So there are two types of improvement I can think of. One is a major, one is a minor. We can always work with minor improvements throughout the iterations. For that, we don't need to change the requirement. But for the major, you know, changes where something has changed in the requirement and for that, you are changing the entire module of software. That takes a lot of effort and time. So I was stressing about that part. But minor changes you can always do, which actually needs to be done because that's what you call the peer review. The peer comes and then sees that somebody's way of commenting is different. Somebody's way of thinking is different. So those kinds of minor changes are acceptable. But I'm a little bit not, you know, a big fan of doing a lot of changes in the middle of the project.

Speaker2: [00:25:05] No, I don't think there's any person who will accept this kind of change. I'm thinking that the Agile methodology will help to some extent to ease this kind of pressure of changing too many requirements. from the very beginning. So yeah, that's what I'm really interested in to understand how these kinds of changes will influence the Agile team's activities or behaviors. So actually, as you mentioned, there are two kinds of changes. For the major changes, it's not very acceptable. But how to solve it is what you want to understand at least understand the background. Why do we need to change that? Right? So that kind of information alignment would be very important and also the flexibility. You also think that's important to implement the changes for self-learning, self-organizing or self-improvement and etcetera. I totally agree with you. I think that the key value of Agile is really to try to learn something from past experiences. And finally, do you have any other comments regarding this kind of agile topic?

Speaker1: [00:26:41] So I can really say that this is a nice way of working. Of course, it's a completely different way. Every process has its pros and cons, but this gives a lot of flexibility, which I really liked it. And the way we are dragging and dropping in the board, that visual looks and initially struggled a lot, but now it's very appealing. And also, a lot of things like the test plans, protocols, and all the options are there in the left pane. It makes everything in one place. So you go to one shop and you get everything, whatever you need. This is really nice. And, of course, the way you are handling the stand-up meetings and all the Agile-related calls, it's really perfect. So that is very nice.

Speaker2: [00:27:30] Okay. Thank you so much. I think that's almost the question I like to ask. And thank you again for this kind of interview. Thank you so much. Okay. Thank you. Bye.