UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rafael da Camara Figueredo

Tailoring agile practices in distributed large-scale contexts: TARGET Framework

Recife

2023

Rafael da Camara Figueredo

Tailoring agile practices in distributed large-scale contexts: TARGET Framework

The master thesis presented to the academic master program of Computer Science from the Federal University of Pernambuco as a partial requirement to obtain the Master's title.

Concentration area: Software Engineering and Programming Languagues

Supervisor (a): Hermano Perrelli de Moura

Co-supervisor (a): Marcelo Luiz Monteiro Marinho

Recife

2023

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

F475t Figueredo, Rafael da Camara

Tailoring agile practices in distributed large-scale contexts: TARGET framework / Rafael da Camara Figueredo. – 2023.

255 f.: il., fig., tab.

Orientador: Hermano Perrelli de Moura.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.

Inclui referências e apêndice.

1. Engenharia de software. 2. Desenvolvimento de software. 3. Agilidade. I. Moura, Hermano Perrelli de (orientador). II. Título.

005.1 CDD (23. ed.) UFPE - CCEN 2023-149

Rafael da Camara Figueredo

"Tailoring agile practices in distributed large-scale contexts: TARGET Framework"

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração Engenharia de software e Linguagens de Programação.

Aprovado em: 19/06/2023

BANCA EXAMINADORA

To my family, my wife, and my professors that supported me in this entire journey . . .

ACKNOWLEDGEMENTS

First, I would like to thank all my family, who believed in me since the beginning and supported and encouraged me in everything I have been doing. I want to thank my mother, brother, brother-in-law, and wife for all the support and help I needed before and during the master's course. Also, I would like to thank my dad, who passed away in 2009 but had never forgotten to dedicate all his life to his children and wife.

I am grateful for my brother, who is truly an inspiration for me to start my master's program. He is the person that most supported and encouraged me to follow my dreams, which indirectly proved to me that we could achieve whatever we want.

I am also thankful for my wife, Larissa, that is with me since the beginning of the course. Without her, this journey would be much more challenging. Thus having a person by my side is extremely important to make me achieve my goals. I want to thank her for putting up with me during weekends and extended work nights, giving me all the support I needed. I promise to make up for everything during our life.

Also, I would like to thank the most inspiring professors I have had during college and who continued with me during the master's course, Marcelo and Suzana, who supported me through this research. Without their support, I would never achieve the results of this project.

To Professors Marcelo and Hermano, my advisors. I want to thank both for being extremely committed to everything related to this research. Also, I would like to thank them for being true inspirations in the research field, teaching me, and constantly elevating me to become a better researcher.

I am also thankful for all the professors of the computer center department of the Federal University of Pernambuco that had contributed to my master's. Specifically, I would like to thank Ivaldir, who was closer to me at the beginning of the research, driving me with Marcelo and Hermano to the correct research path.

To all my friends, the ones I have made during my time at the University, the ones from the graduation who joined the master's with me, and the ones who accompanied my journey, Amanda, Rafael, Paulo, Felipe, Thiago, Samantha, Fábio, Felipe, Nycolas, Bastos, and Luizinho. I want to thank all of them for being present with me in the good and bad moments and in many disciplines. Surviving in both universities was much easier surrounded by them.

Finally, I thank everyone who somehow helped accomplish my master's course.

.

“Persistence is the path to success.”

(CHAPLIN, 1960).”

ABSTRACT

With the increasing adoption of agile methodologies in distributed software development teams, there is a need to adapt these practices for large-scale environments. However, the lack of specific guidance can make this process difficult. This study evaluates how large-scale agile distributed teams are tailoring their practices to accommodate the needs of their context. A Systematic Literature Review Systematic Literature Review (SLR) was conducted on the adaptation of agile in large-scale DSD projects, which identified 95 adapted practices from five agile frameworks used by various case studies between 2007 and 2021. Most studies adapted Scrum with 32 customized practices for their distributed realities in a large-scale context, followed by Scaled Agile Framework Scaled Agile Framework (SAFe) with 24 practices, Large Scale Scrum Large Scale Scrum (LeSS) with 17, the Spotify model with 13, and Disciplined Agile Delivery Disciplined Agile Delivery (DAD) with nine practices. Based on the content of the SLR, it was possible to generate the Tailoring lARge-scale aGilE pracTices Framework (TARGET) framework, which serves as a guide specializing in adapting agile practices to the specific context of various organizations from 17 business domains across different scale dimensions and using different agile and scaling agile frameworks. Finally, a case study was conducted on a distributed agile team from a software consultancy in Brazil to analyze the adherence of the TARGET framework to the reality of a real market case, with positive results.

Keywords: agile; distributed Software development; large-scale; tailoring agile; scaling agile frameworks; agile frameworks.

RESUMO

Com o aumento da adoção de metodologias ágeis em equipes distribuídas de desenvolvimento de software (DSD), surge a necessidade de adaptar essas práticas para ambientes de grande escala. No entanto, a falta de orientações específicas pode dificultar esse processo. Este estudo avalia como as equipes distribuídas ágeis de grande escala estão adaptando suas práticas para acomodar as necessidades do seu contexto. Foi realizada uma Revisão Sistemática da Literatura RSL (RSL) sobre a adaptação do ágil em projetos DSD de grande escala, que identificou 95 práticas adaptadas a partir de cinco frameworks ágeis usados por vários estudos de caso de 2007 até 2021. A maioria dos estudos adaptou o Scrum com 32 práticas customizadas para suas realidades distribuídas em um contexto de larga escala, seguido pelo SAFe com 24 práticas, LeSS com 17, o modelo Spotify com 13 e o DAD com nove práticas. Com base no conteúdo da RSL, foi possível gerar o TARGET, que serve como um guia especializado na adaptação das práticas ágeis para o contexto específico de várias organizações de 17 domínios de negócios, sobre diferentes dimensões de tamanho e usando diferentes frameworks ágeis e escalonáveis. Por fim, foi realizado um estudo de caso em uma equipe ágil distribuída de uma consultoria de Software do Brasil para analisar a aderência do framework TARGET à realidade de um caso real de mercado, com resultados positivos.

Palavras-chave: agilidade; desenvolvimento de software distribuído; larga-escala; adaptação do ágil; frameworks de escalonamento ágil; frameworks ágeis.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ADSD	Agile Distributed Software Development
AGSD	Agile Global Software Development
APB	Area Product Backlog
APO	Area Product Owner
ART	Agile Release Train
ASD	Agile Software Development
BDD	Behavior Driven Development
CD	Continuous Delivery
CI	Continuous Integration
CoP	Community of Practice
DA	Disciplined Agile
DAD	Disciplined Agile Delivery
DoD	Definition of Done
DoE	Definition of Entry
DoR	Definition of Ready
DSD	Distributed Software Development
GSD	Global Software Development
KPI	Key Performance Indicator
LeSS	Large Scale Scrum
MVP	Minimum Viable Product
OKR	Objectives and Key Results
PB	Product Backlog
PI	Program Increment
PL	Product-line
PO	Product Owner

PoC	Proof of Concept
PPM	Program and Portfolio Management
PPO	Proxy Product Owner
RA	Requirement Area
RSL	RSL
RTE	Release Train Engineer
RUP	Rational Unified Process
SAFe	Scaled Agile Framework
SLR	Systematic Literature Review
SM	Scrum Master
SoS	Scrum of Scrums
TAR	Technical Area Responsible
TARGET	Tailoring lARge-scale aGilE pracTices Framework
XP	eXtreme Programming
XSBD	eXtreme Scenario-based design

CONTENTS

1 INTRODUCTION

Since before the pandemic of COVID-19, companies have been strongly pursuing Distributed Software Development (Distributed Software Development (DSD)) (PAASIVAARA; LASSENIUS, 2016; GUPTA; JAIN; SINGH, 2018), among other things, to achieve lower production costs and increase the pool of available, talented individuals (RAZZAK et al., 2018). Meanwhile, after the COVID-19 pandemic, most software industries felt forced to work in a distributed way (CAMARA et al., 2020). This made different organizations look forward to understanding how to manage their software solutions' development across geographically dispersed teams.

As well as DSD has been shaping the form of developing software nowadays, organizations have been looking around to manage the challenges from distributed software development by using agile methodologies that had become the mainstream process for software development, including the distributed (HANSSEN; ŠMITE; MOE, 2011; CAMARA et al., 2020). According to the 16th Annual State of Agile Report, the survey by Digital.Ai, 80% of the respondents say their organizations are working with Agile Software Development (ASD) teams distributed geographically (Digital.Ai, Inc., 2023).

Even with such a presence of agile in the DSD environment, which literature calls Agile Distributed Software Development (ADSD) (PAASIVAARA; LASSENIUS, 2014) or Agile Global Software Development (AGSD) (CAMARA et al., 2020), combining those norms gathers challenges. DSD challenges permeate coordination, communication, and collaboration issues due to the team members' physical distance and time zone differences (MARINHO; NOLL; BEECHAM, 2018; CARMEL; AGARWAL, 2001; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008). Meanwhile, agile methods were originally developed for co-located teams (BECK et al., 2001), which require adaptations for distributed scenarios. However, agile methods appear viable for implementing DSD since it is increasingly used in software development worldwide to achieve faster time to market, quality assurance, and continuous delivery (RAMESH et al., 2006; RAZZAK et al., 2017; MARINHO et al., 2019; PAASIVAARA; LASSENIUS, 2010; GUPTA; VENKATACHALAPATHY; JEBERLA, 2019).

The usage of agile in DSD can not limitate itself to the purest use of agile practices. Organizations with distributed agile teams are usually involved in large-scale projects (DINGSØYR; FÆGRI; ITKONEN, 2014), which require the tailoring and scaling of agile methods to handle the dynamic environment (CAMPANELLI; PARREIRAS, 2015; FITZGERALD et al., 2013a). According-ing to Digital.Ai 16th report, while 80% of the respondents use agile in the development process, 50% of them are gathering agile, traditional, and other iterative approaches, while 35% of them are using a combination of agile frameworks to handle the dynamic scenario of large-scale distributed projects.

The scaling agile frameworks have emerged in response to the difficulty of carrying standard agile methods into large-scale projects (DYBÅ; DINGSØYR, 2008), and their use was intensified since the COVID-19 pandemic has led many teams to work remotely (MAREK; WIŃSKA; DĄBROWSKI, 2021). The agile frameworks promise to cover the gap that agile methods are for co-located teams by guiding companies in applying agile in large-scale and distributed contexts. The most common frameworks, according to Digital.Ai's last Agile report (Digital.Ai, Inc., 2023), are SAFe (Leffingwell, Dean, 2023) with 53% of respondents using it, followed by Scrum@Scale and Scrum of Scrums with 28% (Sutherland, Jeff and Brown, Alex, 2021), Spotify model with 7%, LeSS with 6%, and DA and Nexus with 3%. Moreover, a third of the respondents using a combination of those frameworks say the combinations are not working well, without satisfying the needs of the teams and organizations (Digital.Ai, Inc., 2023).

Most of the dissatisfaction with agile methods and agile frameworks in distributed large-scale environments is related to the fact that those approaches can not be used straightforwardly (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; EWUSI-MENSAH, 2003). Instead, to fully appreciate its benefits, agile practices must be modified and tailored to accommodate the specific context needs of individuals, teams, and organizations (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b), i.e., the general agile ceremonies that usually occur through face-to-face events need to be held through teleconference tools (BASS, 2012; PAASIVAARA; LASSENIUS, 2016; GUPTA; MANIKREDDY; ARYA, 2017; NYRUD; STRAY, 2017), since the team members are not present in the same physical space.

The agile adoption and adaption process is complex and will require effort from every sphere, from the individual to the organization and teams (CAMPANELLI; PARREIRAS, 2015), especially in distributed large-scale companies (ULUDAG et al., 2019; ROLLAND; MIKKELSEN; NÆSS, 2016; EDISON; WANG; CONBOY, 2022). Several authors have already expressed that the one-size-fits-all agile approach may not work since each large-scale project has its particularities and needs (TENDEDEZ; FERRARIO; WHITTLE, 2018; LINES; AMBLER, 2019). Method tailoring must be considered to facilitate the software development process and better suit the context differences among development teams (CAMPANELLI; PARREIRAS, 2015). Agile distributed teams and organizations must consider their unique needs and constraints when selecting and implementing agile practices.

Agile tailoring in software development can be described as adapting agile practices, methods, principles, and values to the organization's aspects, culture, objectives, environment, and context to fit its needs (SALAMEH; BASS, 2020; CAMPANELLI; PARREIRAS, 2015). An agile tailored approach can enhance project value, minimize risks and uncertainties associated with the project's context, and improve team performance and adaptability.

Since organizations have different contexts, necessities, and cultures, what works for one project may not apply to others. Based on it, some authors strongly criticize agile and scaling agile frameworks that summarize best practices, events, and ceremonies while avoiding context understanding and the uniqueness of the organizations and teams (AMBLER; LINES, 2012; LINES; AMBLER, 2019). Such an approach hinders the potential of individuals, teams, and organizations to tailor their practices by narrowing their vision to a set of suggested techniques that do not consider context on its applicability. However, in the literature, it is possible to see few organizations that go beyond collecting techniques from those agile frameworks and look forward to tailoring their agile practices to better suit their day-to-day challenges, issues, and obstacles on large-scale environments with distributed teams (ULUDAG et al., 2019; PAASIVAARA, 2017; RAZZAK et al., 2018; SALAMEH; BASS, 2019; SALAMEH; BASS, 2020; LAL; CLEAR, 2018; GUPTA; MANIKREDDY; ARYA, 2017).

Despite few studies describing agile tailoring through agile and scaling agile frameworks involving distributed teams in large-scale environments, the research field still needs more academic attention. The literature still lacks a deeper and more structured overview of how those ADSD teams tailor the agile practices from those different agile frameworks to handle large-scale settings.

Such discussion regarding the agile tailoring through agile frameworks by DSD teams in large-scale settings launch our study to the following research questions: "What agile practices do the DSD teams that apply agile and scaling agile frameworks in large-scale settings are tailoring?", and "How do DSD teams that apply agile and scaling agile frameworks tailor its agile practices in large-scale settings?". To do so, the present study was divided into three major steps to gather the necessary information to generate a solid, tailored agile strategy framework that could help practitioners and researchers.

First, a Systematic Literature Review (SLR) (KITCHENHAM; CHARTERS, 2007) was conducted to collect sufficient information about the tailoring of agile practices by distributed teams in large-scale contexts that uses agile and scaling agile frameworks to be the most representative study regarding the adaptation of agile practices. Considering studies from 2001 to 2021, a total of 74 studies were selected covering studies that tailored agile while using five of the most popular agile frameworks, Spotify (HENRIK; ANDERS, 2012), SAFe (Leffingwell, Dean, 2023), DAD and DA (AMBLER; LINES, 2012; LINES; AMBLER, 2019), LeSS (LARMAN; VODDE, 2016a), and Scrum and its varieties (SCHWABER; SUTHERLAND, 2020; Sutherland, Jeff and Brown, Alex, 2021; Schwaber, Ken and Sutherland, Jeff, 2022). A total of 95 agile practices from those five frameworks were computed, and the several tailored techniques adopted by the organizations among those practices were described. In general, each practice was described following the structure: I) Name - Title of the practice; II) Goal - The aiming of the practice; III) Who - Which roles are supposed to tailor and apply the practice; IV) How - Description of how the teams tailored the practice; V) Context - How studies from different contexts tailored the practice. This structure helps organize the state of the art regarding agile tailoring.

Second, as the main contribution of this study, the TARGET framework - Tailoring lARge-scale aGilE pracTices Framework - was developed. Based on the information gathered from the SLR, it was possible to generate a framework to serve as a guideline for practitioners and researchers to understand better how various organizations with different business domains have tailored agile practices using different agile and scaling agile frameworks already known. Within the extracted information regarding the tailored agile practices from the SLR, it was possible to categorize the findings by classifying the market sectors of the studies fully read, their scale dimensions, and the agile framework in use (DINGSØYR; FÆGRI; ITKONEN, 2014). By categorizing it, a visual structure representing the TARGET framework was developed with 17 market sectors covering different scale dimensions and agile frameworks that gather several tailored agile practices which anyone can consult with a similar dimension or market sector from the organizations assessed.

Third, an evaluation step was also established. A case study was settled in an agile distributed team from an IT service provider company in Brazil with more than one thousand employees to evaluate the TARGET Framework better. By assessing an agile distributed team with eight members spread across four out of the five regions of Brazil and a quality tech leader working on a very large-scale project with more than twenty distributed agile teams. The case study consisted of a single assessment of a sample of employees from the same context but that work on different modules of the same very large-scale solution. The case study aims to evaluate the presence of tailored agile practices identified in the literature by assessing an agile remote distributed team from an IT service provider company. By doing it, we hope to identify similarities among the tailored practices from one of the market sectors of the TARGET framework used by the members in the case and their different tailored approaches in adapting agile practices.

This study presents results regarding agile tailoring in distributed teams using agile and scaling agile frameworks from large-scale settings. Based on the analysis, this study contributes through the development of a reference framework that benefits the academy and industry through a complete overview of how several organizations from 17 different market sectors and different scale dimensions using different agile frameworks are tailoring their agile practices to continue the development and continuous delivery of value. Also, by summarizing the agile practices tailored by organizations from different domains and describing how different contexts had tailored those practices. Further, organizing the information in a structured view through the TARGET framework and contributing to the agile tailoring research by evaluating the developed framework in a real industry case in an IT company from Brazil, an under-researched region in the agile tailoring field. Finally, in further sections, it will be possible to see some small-scale studies that became part of the TARGET Framework since it describes challenges and environments from DSD and large-scale projects (DINGSØYR; FÆGRI; ITKONEN, 2014).

Towards the findings from the literature, we hope to provide the TARGET framework as a collection of techniques to improve the agile practices used by organizations among distributed development teams. Similar to the strategy proposed by Ambler and Lines in choosing your way of working (LINES; AMBLER, 2019), we hope to provide a context-specific guide to help companies during the agile tailoring process according to their uniqueness. Finally, despite the evaluation in a specific market sector, IT service companies, the framework covers the organization's needs from distinct market sectors.

The remainder of this study is organized as follows: In chapter 2, we introduce the background regarding the research subjects that explains the research problem. Chapter 3 describes the proposed methodologies and the research questions. Chapter 4 present the results and implications from the SLR, the TARGET framework, and the case study. Then, chapter 5 has the purpose of discussing the findings and limitations. Finally, in Chapter 6, we state some concluding remarks and areas of future research directions.

1.1 OBJECTIVES

The specific goals of this study are twofold. First, we aim to develop the TARGET framework to help organizations tailor agile practices in distributed large-scale projects. Second, we aim to evaluate the adherence to this framework in practice by evaluating it in a real-world case study. By achieving these goals, we hope to contribute to the growing body of knowledge on agile, DSD, and large-scale development and provide practical guidance for organizations seeking to adopt agile practices at scale.

On the other side, The specific goals of this study are:

- Provide a comprehensive overview of the state of the art of agile tailoring in distributed teams on large-scale projects through an SLR.
- Based on the review findings, develop the TARGET framework, which will summarize a collection of tailored agile practices and ways to implement it for large-scale distributed projects, organized by market sectors, scale dimensions, and agile frameworks used.
- Evaluate the adherence of the TARGET framework in a real-world case study within one of the market sectors described in the framework.

With these goals in mind, this study will provide valuable insights into the potential of the TARGET framework to help agile implementation in large-scale settings with distributed teams.

2 BACKGROUND

This chapter aims to present a brief theoretical foundation about the subjects related to this project. Also, It intends to give the reader an understanding of the research areas. Based on it, the following sections will cover the research areas of ASD, DSD, AGSD, large-scale agile projects, agile tailoring, and related works.

2.1 AGILE SOFTWARE DEVELOPMENT (ASD)

The concept of agile software development emerged in the 1990s (AOYAMA, 1998). However, it gained widespread popularity following the publication of the agile manifesto in 2001 (BECK et al., 2001). The increasing business need for fast creation of the internet and mobile applications was a key driver for introducing these lightweight and nimble development processes (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). The agile ideas promised that higher customer satisfaction can be achieved by addressing such uncertainty aspects and delivering working software frequently with shorter timescale (WAHYUDIN et al., 2008).

The ideas of agile software development have gained acceptance in the mainstream software development community. Surveys pointed out that agile teams are often more successful than traditional ones. According to the agile manifesto (BECK et al., 2001), ASD emphasizes individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.

The most widely used methodologies based on agile principles are Scrum and eXtreme Programming (XP) (Jalali; Wohlin, 2010). However, other methods such as Dynamic Systems Development Method, Adaptive Software Development, and the Crystal Family stress upon short time goals and incremental delivery, dividing the entire projects into sprints and every sprint governed by a complete software development life cycle (Sriram; Mathew, 2012).

The success of ASD depends significantly on team interaction (DORAIRAJ; NOBLE; MALIK, 2012). Agile methods have enabled software project teams to meet the challenges of an even more turbulent business environment through enhanced flexibility and emergent customer needs (MARUPING, 2010).

Kruchten (KRUCHTEN, 2013) defines agility as "the ability of an organization to react to changes in its environment faster than the rate of those changes". This definition uses the ultimate goal of being agile for business rather than defining agility by a set of labeled practices (for example, you are agile when running XP (BECK; GAMMA, 2000), Scrum (Schwaber, Ken and Sutherland, Jeff, 2022), or Lean (POPPENDIECK; POPPENDIECK, 2007)) or by a set of properties defined as opposed to another set - the agile manifesto (BECK et al., 2001). This definition is not far from Conboy's, which is addressed in his research on the literature on agile process development (CONBOY et al., 2011).

2.2 DISTRIBUTED SOFTWARE DEVELOPMENT (DSD)

Distributed Software Development (DSD) is the name to designate software projects developed by teams spread beyond the company's boundaries and in different geographical locations (MARINHO et al., 2019). Such boundaries could be national, which describes DSD, or international, which refers to Global Software Development (GSD) (MARINHO; NOLL; BEECHAM, 2018). From a global perspective, teams are distributed across different countries or continents while working as virtual teams, together online from different locations (MARINHO; NOLL; BEECHAM, 2018).

Globalization has made this software development approach more popular in the last decade since it has become possible to have members worldwide working through remote tools. It also enables organizations to access highly skilled talents from around the globe, reducing costs, seeking a 24-hour development cycle due to different time zones, and accessing low labor costs (ZAHEDI; SHAHIN; Ali Babar, 2016; MARINHO et al., 2019).

Despite the benefits of developing software in a distributed manner, those benefits do not come for free. Most organizations must deal with several challenges related to inadequate communication, cultural differences, team coordination, timezone differences, and coordination difficulties (MARINHO; NOLL; BEECHAM, 2018).

Most DSD challenges can be categorized into three significant distances that impact those teams. The geographical, cultural, and temporal distance (CARMEL; AGARWAL, 2001).

- **Geographical distance:** as the name implies, the geographical distance is related to the physical separation among the members of a distributed software team. It can cause communication delays and travel costs and is also can be the origin of the next distance;
- **Temporal distance:** due to the physical dispersion of the development teams, the

temporal distance can be felt based on different timezones. Such distance will depend on the degree to which the team members work synchronously or asynchronously, which can be minimized by defining a common overlap work hour among the members;

- **Cultural distance:** it can manifest as organizational and national cultures. The organization's culture relies on norms, values, principles, and rules varying from small to large companies. Meanwhile, the national distance relies on the different languages, ethnic groups, customs, and beliefs of the team members. Such distance can cause communication barriers and misunderstandings affecting collaboration.

To address the challenges from distributed development and handle the issues that emerge from the three major distances of distribution's nature. The teams and companies must be aware to provide an environment that foments effective communication, collaboration, and the necessary practices to achieve the project's success (CAMARA et al., 2020). To achieve this, many organizations are adopting agile methods to manage the obstacles the distribution settings can cause (CAMARA et al., 2020).

2.3 AGILE GLOBAL/DISTRIBUTED SOFTWARE DEVELOPMENT (AGSD/ADSD)

The origins of ADSD can be traced back to the early 2000s (JALALI; WOHLIN, 2012), just a few years after the introduction of the Agile Manifesto (BECK et al., 2001). "Agile Global Software Development" (AGSD) or "Agile Distributed Software Development" (ADSD) refers to the application of agile practices, methodologies, and techniques on globally distributed projects (PAASIVAARA; LASSENIUS, 2014).

Since then, studies have shown that using agile practices is directly related to the success of DSD projects (TENDEDEZ; FERRARIO; WHITTLE, 2018; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008) reported that large and distributed software development projects that face challenges related to technologies and requirements could benefit from using agile practices to improve organization and management.

However, achieving success in a DSD project is a challenging task, which requires effective planning, organization, leadership, control, coordination, and project management (CAMARA et al., 2020). These practices aim to mitigate the challenges of geographic, temporal, and socio-cultural distances (CARMEL; AGARWAL, 2001). While agile practices were initially developed to help co-located teams (VALLON et al., 2017), numerous studies demonstrate that agile methods can help mitigate DSD problems (TENDEDEZ; FERRARIO; WHITTLE, 2018; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). Therefore, agile methods appear to be a viable option for implementing DSD as it is increasingly used in software development globally.

To adapt to the challenges of DSD, practices such as daily meetings, planning, and pair programming are tailored (CARMEL; AGARWAL, 2001; BASS, 2012). For instance, face-to-face meetings and pair programming cannot be implemented the same way as for co-located teams since team members are in different locations and possibly in different time zones (RAJPAL, 2018; HODA et al., 2010). Thus, teams must choose asynchronous and synchronous communication tools to implement these practices, such as video conferencing, instant messaging, emails, and other collaboration tools (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; LEE; JUDGE; MC-CRICKARD, 2011; HOSSAIN; BABAR; VERNER, 2009).

In the context of AGSD and ADSD, Scrum is the most widely used framework, followed by Kanban (MARINHO et al., 2019). It is common to observe traditional DSD projects transitioning to agile approaches to reduce the complexities of a distributed context (GUPTA; JAIN; SINGH, 2018) and improve collaboration and organization of the development process.

In summary, AGSD and ADSD are software development approaches that combine Agile practices and principles with distributed software development challenges. Many companies use DSD on large-scale settings to access global talent, reduce expenses, gain economic advantages, achieve faster delivery, and enable a 24-hour software development cycle due to different time zones (RIZVI; BAGHERI; GASEVIC, 2015). However, while adopting distributed development has increased in companies, having team members in different locations can result in communication, coordination, and control problems in the development process (BASS, 2012), which relies upon the teams to tailor their process.

2.4 LARGE-SCALE AGILE PROJECTS

Large companies constantly looking to distribute their development are often handling large-scale projects. Sometimes, those large-scale projects are complex and belong to critical sectors (CHO, 2007). A large-scale project also requires development methodologies to scale and fit the environment settings (DINGSØYR; FÆGRI; ITKONEN, 2014). This is necessary due to the involvement of multiple teams working together on a single product, module, or project in a large-scale setting.

There are discussions over the exact meaning of "large-scale agile", with Dikert et al. (DIKERT; PAASIVAARA; LASSENIUS, 2016) noting that "what is seen as large-scale depends very much on the context and the person defining it". On the other hand, the number of teams is also a metric to determine the project's scale. Due to it, this study follows the taxonomy presented by Dingsoyr *et al.* (DINGSØYR; FÆGRI; ITKONEN, 2014), which defines the scale of agile software development projects based on the number of teams involved and classifies it into three levels of scale.

- **Small-scale:** those projects usually have only one team, and the coordination may require the use of regular agile practices without any proper tailoring approach.
- **Large-scale:** those projects can have 2 to 9 teams, and the coordination will need scaling approaches, such as Scrum of Scrums.
- **Very large-scale:** whether the project has ten or more teams, it can be considered a very large-scale project. For such projects, the use of a scaling framework is recommended.

Adopting agile methodologies in large-scale agile projects has gained popularity recently to improve collaboration, communication, and innovation while delivering high-quality software products on time and within budget, (PAASIVAARA, 2017; GUPTA; VENKATACHALAPATHY; JEBERLA, 2019; PAASIVAARA et al., 2014). Moreover, it also encourages companies to work with multiple teams while being able to adapt faster to changing requirements and market conditions (DANEVA et al., 2013).

Implementing agile methodologies in large-scale agile projects can also present challenges. For example, the larger the team, the more difficult it can be to coordinate and synchronize efforts, and the more complex the project management becomes (LARMAN; VODDE, 2016b). Furthermore, the cultural and organizational changes required to adopt agile methodologies can be significant, requiring a commitment from all stakeholders, senior management, directors, and sometimes the board and a willingness to embrace new ways of working by every interested part (MATTHIESEN; BJØRN, 2017; PAASIVAARA et al., 2014; DORAIRAJ; NOBLE; ALLAN, 2013). To achieve such a level of commitment during agile adoption, many companies rely on adopting scaling frameworks (PAASIVAARA, 2017; SALAMEH; BASS, 2019; PANDYA; MANI; PATTANAYAK, 2020).

2.5 SCALING AGILE

Scaling agile applies agile methodologies to large and complex environments involving multiple teams, departments, and stakeholders (Denning, Steve, 2021). The most common approach to scaling agile is to use a framework that provides guidelines and best practices for coordinating the work of multiple teams. Several frameworks can be used to scale agile, each with its strengths and weaknesses.

Initially designed for small co-located teams, traditional agile methods struggle to work in these large-scale environments. The need for coordination, collaboration, and alignment across multiple teams, departments, and stakeholders can become a significant challenge that traditional agile methods can not handle (DYBÅ; DINGSØYR, 2008; FITZGERALD et al., 2013b).

To address this challenge, several scaling agile frameworks have emerged (HENRIK; ANDERS, 2012; Leffingwell, Dean, 2023; AMBLER; LINES, 2012; LARMAN; VODDE, 2016a; Sutherland, Jeff and Brown, Alex, 2021; Schwaber, Ken and Sutherland, Jeff, 2022). These frameworks provide guidelines and best practices for coordinating the work of multiple teams, improving communication and collaboration, and ensuring alignment with business goals.

According to Scott Ambler (AMBLER; LINES, 2013), scaling agile can be interpreted in three ways. Firstly, scaling agile through the organization involves scaling agile across teams, requiring a high level of collaboration and understanding of business value among teams. Secondly, scaling agile for other projects consists of scaling the project's size, which can be challenging due to the need to find specialized people for the project. Thirdly, scaling agile to the entire value stream of an organization involves scaling and maximizing the development of business value without focusing solely on agile teams and coordination.

Founder of Agile For All, Hartman (Denning, Steve, 2021) proposes two concepts for scaling agile - Horizontal and Vertical. Horizontal scaling involves applying agility concepts in other organization sectors to make all the processes agile, while vertical scaling agile aims to impact the director board by coordinating all sectors, corporate governance, and shareholders' representations. According to Hartman, the goal of an organization looking to scale agile should impact both horizontal and vertical ways to achieve a fully agile organization. However, this continuously evolves and cannot be achieved overnight.

The most recent State of Agile Report survey (Digital.Ai, Inc., 2023) indicates that the Scaled Agile Framework (SAFe) (Leffingwell, Dean, 2023) remains the most popular scaling framework, with adaptations of Scrum (Schwaber, Ken and Sutherland, Jeff, 2022) for large-scale settings, such as Scrum@Scale (Sutherland, Jeff and Brown, Alex, 2021) and Scrum of Scrums, following closely behind. Although less prevalent, other scaling frameworks, such as Disciplined Agile Delivery (DAD) (AMBLER; LINES, 2012), Large-Scale Scrum (LeSS) (LARMAN; VODDE, 2016a), and the Spotify model (HENRIK; ANDERS, 2012), are also being used.

On the road to facilitating agile insertion in globally distributed teams, many companies choose those frameworks to scale agile practices. In the following lines, we will briefly describe five of the most common scaling agile frameworks (ALQUDAH; RAZALI, 2016). From those five frameworks, we will consider Scrum and its adaptations of large-scale settings as one of them.

Scaled Agile Framework (SAFe) - The SAFe (Leffingwell, Dean, 2023) was developed by Dean Leffingwell in 2012, and it focuses on scaling agile on large enterprises. The framework is a documented and proven approach to scaling agile practices, strategies, and benefits in large enterprise environments. Currently, the SAFe is in its 6.0 version (Leffingwell, Dean, 2023) and was developed to help companies manage, control and organize the development process in a context with many teams and people (RAZZAK et al., 2018; PAASIVAARA, 2017).

SAFe 6.0 framework is a solid structure that covers all organization levels. Its four core values alignment, built-in quality, transparency, and program execution (Leffingwell, Dean, 2023) is associated with values and principles from Scrum (Schwaber, Ken and Sutherland, Jeff, 2022), eXtreme programming (BECK; GAMMA, 2000), Lean Software Development (POPPENDIECK; POPPENDIECK, 2003) and the Agile Manifesto (BECK et al., 2001). Recently, SAFe 6.0 was launched (Leffingwell, Dean, 2023). This new release aims to cover all enterprise levels and enable the digital business needed since the pandemic of COVID-19 forced. Furthermore, the latest version brings six primary themes into evidence: I) Strengthening the Foundation for Business Agility; II) Empowering Teams and Clarifying Responsibilities; III) Accelerating Value Flow. IV) Enhancing Business Agility with SAFe across the business; V) Building the Future with AI, Big Data, and Cloud; VI) Delivering Better Outcomes with Measure and Grow and OKRs.

Large-Scale Scrum (LeSS) - The Large-Scale Scrum (LeSS) (LARMAN; VODDE, 2016a) is a framework developed by Larman and Vodde. It has two approaches: (i) The LeSS framework is designed for medium companies, around 70 people within five teams and two different sites; and (ii) LeSS huge designed for large companies with more than eight teams, that may work as a scaled LeSS for several smaller fewer applications. In essence, LeSS is Scrum fitted for larger and more complex projects without losing its principles, combining Lean and XP practices (LARMAN; VODDE, 2016a).

The framework is based on transparency, inspection, and adaptation principles since it is based on Lean and encourages teams to work together to develop the best possible product. One of the critical features of LeSS is that it emphasizes simplicity and avoids unnecessary complexity, which can lead to confusion and reduce the framework's effectiveness (LARMAN; VODDE, 2016a). Less is also designed to be flexible and adaptable to the needs of different organizations, and it provides guidance on how to customize the framework based on its boundaries to meet the specific needs of a particular project or organization (LARMAN; VODDE, 2016a).

Disciplined Agile Delivery (DAD) - Disciplined Agile Delivery (AMBLER; LINES, 2012) is a toolkit developed by Ambler and Lines. The framework extends Scrum practices and combines them with other agile methods, such as XP, Lean, and Kanban. The framework covers approaches from the beginning of a team formation until delivery and production support (AMBLER; LINES, 2012). The DAD is designed for enterprise-level projects, which may need to organize multiple teams and their efforts and work through a structured governance framework.

DAD has a robust governance structure that defines roles and responsibilities, provides guidance on decision-making, and ensures compliance with organizational standards and policies (AMBLER; LINES, 2012). DAD's process comprises three phases: Inception, Construction, and Transition, each with activities, artifacts, and guidelines. The Inception phase defines the project start-up by starting requirements modeling, initial architecture modeling, scope, team formation, and stakeholders identification, and develops an initial plan for development and release. The Construction phase involves iterative development and testing of the software to build the deliverables of the solution. Meanwhile, the Transition phase focuses on deploying the solution and ensuring it meets the stakeholders' needs (AMBLER; LINES, 2012).

DAD has evolved, and the DA toolkit has become an evolution of the original model. The new DA toolkit expands on DAD's original principles and practices by incorporating context-sensitive guidance rather than being a collection of best practices from agile methodologies (LINES; AMBLER, 2019). DA allows teams to tailor their approach to their specific needs, selecting the most appropriate process and practices that will work best in their context, which will describe their way of working (LINES; AMBLER, 2019). In addition, DA recognizes that teams must work within their organization's way of working and can adopt and adapt different frameworks and practices, including Scrum, Kanban, Lean, SAFe, and others, depending on what works best for them (LINES; AMBLER, 2019). For the authors, DA emphasizes a pragmatic, context-based, and enterprise-aware approach to agile and lean delivery (LINES;

AMBLER, 2019). Different from regular agile frameworks that avoid context understanding and commonly provide a narrowly set of best practices without considering the uniqueness of individuals, teams, and organizations.

Spotify - The Spotify model (HENRIK; ANDERS, 2012) was developed by Henrik and Anders while they were working on the Spotify company. The Spotify model is based on the agile principles of cross-functional teams, autonomy, and continuous improvement (HENRIK; ANDERS, 2012). The model is designed to enable organizations to adopt agile practices and principles and scale them to a larger organization. The Spotify model is not a prescriptive framework with a collection of processes, tools, and best practices but rather a set of principles and concepts that an organization can customize and adapt to suit its needs. It organizes the organization's developments into the model's key features, the squads, tribes, chapters, and guilds. Each squad uses its preferred agile practices, which could be a combination of Scrum, Kanban, Lean, or other agile methodologies.

A squad is a self-organized and multi-disciplinary agile team with 6-12 members responsible for developing a product or a module of it (HENRIK; ANDERS, 2012). Moreover, squads are organized into tribes, representing a group of squads working on related areas (HENRIK; ANDERS, 2012). Chapters are cross-functional communities of practice that allow individuals in similar roles across squads to share knowledge and expertise (HENRIK; ANDERS, 2012). Guilds are communities of interest that enable individuals to come together and share knowledge and expertise across tribes and chapters (HENRIK; ANDERS, 2012).

Numerous organizations have adopted the Spotify model, but some found it challenging to implement due to the high degree of autonomy demanded by the teams. Due to it, it is necessary to adapt the model according to the context of the organization (SALAMEH; BASS, 2019). Regularly, the Spotify model would fit better for companies with similar business models of the Spotify company. However, it is essential to point out that the known model is a "screenshot" of how Spotify worked in 2012. Nowadays, the company may apply a different approach.

Scrum - The Scrum methodology was originally developed for co-located teams working in the same physical space. Ken Schwaber and Jeff Sutherland, in the 1990s, introduced it. It has since gained popularity at the beginning of the century due to its simplicity, flexibility, and focus on delivering aggregate value to customers (Schwaber, Ken and Sutherland, Jeff, 2022). Scrum is based on the principles of transparency, inspection, and adaptation. Scrum defends the building of cross-functional teams within 7-9 team members working together to deliver a potentially shippable product increment at the end of each iteration, described as sprint (Schwaber, Ken and Sutherland, Jeff, 2022).

In its pure way, Scrum is not a scaling framework. But, by being the most popular Agile methodologies according to Version 16th survey state of agile (Digital.Ai, Inc., 2023). We will consider its adaptions, and tailored approaches for large-scale scenarios, such as Scrum@Scale (Sutherland, Jeff and Brown, Alex, 2021) and Scrum of Scrums (PAASIVAARA; LASSENIUS; HEIKKILä, 2012).

Scrum@Scale and Scrum of Scrums (SoS) are two Scrum framework adaptations aiming to scale agile practices in large settings. The Scrum@Scale (Sutherland, Jeff and Brown, Alex, 2021) is a framework that aims to provide scalable and flexible ways for organizations to manage multiple teams. The framework is based on several practices, artifacts, and agile events tailored for the scaled environments of large organizations. By default, the framework has three components: I) The Scrum Master (SM) Cycle: Coordinating the "How"; II) The Scrum Product Owner (PO) Cycle: Coordinating the "What" and; III) Connecting the PO and SM Cycles.

Moreover, Scrum of Scrums is a technique to coordinate multiple Scrum teams working in the same value stream. The SoS is also a tailored practice to adapt daily meetings across several Scrum teams (PAASIVAARA; LASSENIUS; HEIKKILä, 2012). The SoS meeting gathers the Scrum Master of each Scrum team for a daily of Scrum Masters. Each Scrum Master has the duty of o discuss the status of each team's work, dependencies, and issues that might impact the overall project progress (PAASIVAARA; LASSENIUS; HEIKKILä, 2012). The meeting aims to facilitate team communication and coordination to ensure they work towards a common goal.

2.6 AGILE TAILORING

Agile methods were developed for small, co-located teams working in physical spaces (BECK et al., 2001). However, agile practices have been adapted to different contexts since their emergence, from small to large companies and regulated industries to software factories worldwide (DYBÅ; DINGSØYR, 2008).

It was assumed that general software development methods could be applied in any development project and application (CESARE et al., 2008). However, many software project failures have been attributed to the absence of evaluations of development methodologies (EWUSI-MENSAH, 2003). A one-size-fits-all approach could lead organizations to lack flexibility, have inadequate communication, and limited skills to adapt to changes (TENDEDEZ; FERRARIO; WHITTLE, 2018). Due to this, the research community has started to discuss method tailoring to facilitate the software development process and better suit the context differences among development teams (CAMPANELLI; PARREIRAS, 2015).

Research in agile tailoring will likely continue as it enhances team performance, adaptability, and delivery by accommodating practices according to context. Meanwhile, to tailor effectively, organizations may consider several factors, including project complexity, team composition, customer requirements, and organizational culture (HOSSAIN; BANNERMAN; JEFFERY, 2011).

Agile tailoring in the software development process can be classified as the ability to tailor agile practices, principles, and values to create an approach that fits the needs of a project, team, or organization. This tailored approach enhances project value, minimizes risks and uncertainties associated with the project's context, and improves team performance and adaptability. Furthermore, in the literature, Conboy and Fitzgerald defined two classifications for method tailoring (CONBOY; FITZGERALD, 2010; FITZGERALD; RUSSO; O'KANE, 2000): contingency factors and method engineering.

The contingency factors approach for method tailoring considers that the best way to tailor the software development process is by combining the principles, values, and practices of multiple available methods and selecting those that best suit the organization's and teams' needs regarding structure, context, application type, project formalization, and personal preferences (FITZGERALD; RUSSO; O'KANE, 2000). This approach assumes that no formatted framework or methodology can meet the organization's needs, and only a combination of different methods can help them overcome their challenges.

To properly adopt the contingency factors strategy, the company must define a portfolio of frameworks and methodologies so that teams can select the best practices (FITZGERALD; RUSSO; O'KANE, 2000). Additionally, companies must define tailoring criteria for the selection process, and the development context must be considered since it will drive the tailoring process (CAMPANELLI; PARREIRAS, 2015).

The challenging parts of the contingency factors approach are related to combining multiple methods. A set of procedures may not address all contingencies. Applying such an approach also requires a good level of knowledge from the team members regarding many methods, which can demand a lot of training and courses in an environment where changes occur quickly (KUMAR; WELKE, 1992).

On the other hand, the method engineering approach bases itself on building a new method extended from existing method fragments (CONBOY; FITZGERALD, 2010). Software development methods will be created based on the organization's and the teams' specific context, but not by adopting regular practices from several methodologies presented in the community (CAMPANELLI; PARREIRAS, 2015). The main point of method engineering is to tailor the practices adopted to respond directly to the challenges faced by real projects with specific contexts.

The adaptations made by team members allow the teams to be the creators of their method, easing their way of work and bringing the challenge of controlling it among the organization (HENDERSON-SELLERS; RALYTE, 2010).

The process of developing a method using method engineering can be long. It would require a fragment repository and an owner, the method engineer, responsible for configuring the method based on the environment's specific needs (FITZGERALD; RUSSO; O'KANE, 2000). In the process, a continuous cycling and improvement approach must be taken considering the project environment and characteristics, the fragment selection and building, the project performance, the constant evaluation of the practices being applied, and its adaptations to achieve better results (CAMPANELLI; PARREIRAS, 2015).

The method engineering approach also requires a high degree of expertise and knowledge in software development methodologies and practices and significant time and resources to build and maintain the repository of method fragments. Additionally, the success of the method engineering approach depends on the method engineer's ability to understand the specific needs of the project and the team, which can be challenging in complex or rapidly changing environments.

In conclusion, the contingency factors and method engineering approaches have advantages and disadvantages through the agile method tailoring. While the contingency factors approach allows for more flexibility and adaptability by combining multiple methodologies, implementing it can be challenging and requires significant knowledge and expertise (CAMPANELLI; PARREIRAS, 2015). Meanwhile, the method engineering approach can be more tailored to the specific needs of the project and team. Still, building and maintaining the repository of method fragments requires considerable time and resources and relies heavily on the method engineer's expertise and knowledge regarding the organization domain (CAMPANELLI; PARREIRAS, 2015). Finally, organizations must consider their specific needs and contexts when choosing an agile tailoring approach that best fits their needs.

2.7 RELATED WORKS

Hoda *et al.* (HODA et al., 2010) conducted a very large case study in 16 organizations across India and New Zealand, corroborated with four other case studies. The authors aimed to better understand how agile software development teams from those organizations dealt with agility in their project contexts. Since agile methods were defined for small co-located teams in projects with variable scopes, the paper aimed to identify how agility was perceived in contexts different from traditional agile projects. Through 40 interviews with diverse people from different roles, the authors applied a rich ground theory process to categorize the contexts of agility and its adaptation strategies. First, the context of the lack of customer involvement was addressed through story owners, customer proxy members, and simulation. Second, the context of fixed-bid contracts was addressed by providing options as contracts for a batch of sprints and buffering development time. Design/Architecture Intensive was the third context, which was adapted through a design pipeline, information architecture, and walking skeleton. Then, Documentation intensive was adapted through a project dictionary in web projects, but with comprehensive docs in a project with regulated environments. The fifth was the slow rate of change that was tailored through working from requirements, which consists of making iterations through a stable project scope. The last context was about distributed teams that usually adapt communication and collaboration through video and audio conferences or instant chat messages. Further, the authors discuss context-independent practices that teams used from agile methods, followed by the book. Besides it, context-dependent methods that required a level of tailoring were also discussed, such as release planning, backlog refinement, and user story writing. Finally, the study from Hoda *et al.* (HODA et al., 2010) highlights the importance of context during agile adoption and tailoring and how it can impact the team's routine. Since the study was conducted more than ten years ago, we reinforce the need to evaluate new contexts through our SLR to understand how agile is tailored.

Hossain and Bannerman conducted a multi-case study (HOSSAIN; BANNERMAN; JEFFERY, 2011) on four different GSD projects to evaluate the tailoring of regular agile Scrum practices. The authors also assessed contextual factors of GSD projects that may limit the use and tailoring of agile methods. The authors chose seven primary practices of Scrum and described how it was tailored by each one of the four case studies. The agile practices chosen were entirely derived from Scrum, and they were: sprint, sprint planning, daily Scrum, Scrum of Scrums (SoS), sprint review, sprint retrospective, and backlog. Finally, the authors summarize the factors that lead each phase to tailor each agile practice. They also describe in full detail what each factor impacted and the consequence of it. The factors are project size, collaboration modes, number of distributed sites, project domain, budget team size, and others. Despite the focus given to case studies using Scrum, it was perceived that there is no specific way to tailor Scrum in a GSD environment. Probably other agile methodologies and frameworks, too, since different market sectors would have various contextual factors that would imply the tailored approaches used. Based on it, we reinforce the present study's importance in investigating the literature to aggregate the different tailored strategies used by different agile distributed teams in large-scale settings using different agile methodologies and frameworks in various market sectors.

Bass, in his studies (BASS, 2014; BASS, 2013; BASS, 2015), conducted a series of interviews with 46 practitioners from 8 large-scale international companies. During a grounded theory process, he evaluated the tailoring of the Scrum Master (BASS, 2014) and Product Owner roles (BASS, 2013; BASS, 2015) in those large-scale contexts to understand better how the practitioners describe the enhancement and expansion of such a role. The study was conducted with the same companies with offices across the UK, India, Germany, the USA, Malaysia, and New Zealand. Such companies provide services on the Internet, industrial, enterprise, and regular software factories. Various interview members were selected, including developers, QA analysts, Product owners, managers, and even Chief technology officers. The Scrum Master study (BASS, 2014) organizes the data collected in six scrum master activities and how such activities were conducted in those companies. Such activities are: I) Process anchor; II) Stand-up facilitator; III) Impediment remover; IV) Sprinter planner; V) Scrum of Scrums facilitator; IV) Integration Anchor. Conversely, the Product Owner studies (BASS, 2013; BASS, 2015) categorize the data collected in nine product owner functions tailored to scale agile in large projects. Such roles are: I) The groom; II) The prioritizer; III) The release master; IV) The Technical architect; V) The Governors; VI) The communicator; VII) Travelers; VIII) The intermediary; IX) The risk assessor. The author also classifies those POs' roles into two different classes, the client-side POs' more concerned with client-related tasks, and the production-side POs' that would work closer to the technical and legal aspects of the solutions (BASS, 2013; BASS, 2015).

Both studies from Bass (BASS, 2014; BASS, 2013; BASS, 2015) describe in rich detail how two vital roles from Scrum are being tailored in large-scale environments from companies of different market sectors. However, by presenting various activities and functions of the PO and

SM role, the studies are limited to those roles without giving further information on how the agile practices are tailored during the applicability of those different functions. Due to it, we are looking forward to expanding Bass's findings by providing a more extensive overview, not focused on agile roles but on tailored practices in general at large-scale settings.

Campanelli and Parreiras (CAMPANELLI; PARREIRAS, 2015) conducted an SLR study to systematically review the existing literature on agile methods tailoring and summarize and understand how the research on agile tailoring is undertaken. Also, identify the technical aspects and the research community's view of agile methods tailoring, and identify trends and gaps in the existing research. The study presented the concepts of contingency factors and method engineering theory, showing the studies that used each approach. More than that, the analysis demonstrated some criteria that led the reviewed studies to tailor agile methods, such as project type 46.4%; Business goals 42.9%; Complexity 26.8%; Team size 25.0%; Technology knowledge 23.2%; User availability 16.1%; Requirements stability 16.1%; Organization size 16.1%; Culture 16.1%; Team distribution 14.3%; Management support 8.9%; Degree of innovation 7.1%; Previous projects 7.1%; Maturity level 7.1%; Domain knowledge 5.4%; Project budget 5.4%; Communication 5.4%; Type of contract 3.6%. Despite the cover of the study regarding the tailoring approaches of several empirical studies, the analysis does not focus on distributed agile teams in large-scale environments but on agile teams in general. Such a gap highlights the importance of further investigation of agile tailoring in large-scale distributed environments and the criteria related to those settings.

Rizvi *et .al* (RIZVI; BAGHERI; GASEVIC, 2015) conducted an SLR to identify why companies adopt AGSD, which are the most critical risks and threats in AGSD, and which agile methodologies lead AGSD projects to success. The paper selected 63 AGSD studies from 2007 to 2012 that applied practices from all agile methods. The article's primary goal is to understand the reasons for adopting agile in GSD and synthesize the most present risks and threats of distributed projects and, consequently, the best solutions to mitigate them. Also, it aims to relate the agile methodology used with risks faced by the projects and the mitigation strategies used. Unlike our goal, which aims to describe how the AGSD teams from large-scale projects applied agile practices, Rizvi's studies do not explain how those practices were tailored in the studies. Besides, it covered a small range of years of research studies. Finally, it does not consider scaling agile practices to better deal with the risks and threats of large-scale AGSD projects.

Rolland *et al.* (ROLLAND et al., 2016) present a study that examines the underlying assumptions in existing studies of large-scale software development. The authors used the problematization methodology from Alvesson and Sandberg to develop an alternative set of assumptions better suited to the characteristics of large-scale agile software development. The authors also conducted a case study of a large-scale agile project that involved 120 participants in a large company from Norway. As an output of the study, the authors offer a new set of assumptions for "Agile in the large" since how knowledge can be transferred, how inter-team coordination needs to be addressed, and how replacement assumptions are not silver bullets for more traditional teams. Moreover, the study provides critical questions and suggests new avenues for research to help overcome the various barriers that practitioners might face in this area. Based on the author's suggestions and assumptions presented, it was possible to evaluate how agile in the large can assume different shapes according to the organizational context, which is aligned with our beliefs that agile requires specific tailoring techniques according to the teams' context and needs.

In one of his studies, Bass uses grounded theory to explore the process of agile tailoring by investigating artifact inventories, using empirical data collected from industry practitioners at all levels representing nine international companies involved in large-scale offshore software development scenarios from several different market sectors (BASS, 2016a). The study described artifacts related to program governance, product artifacts, release artifacts, Sprint artifacts, and future artifacts categories. During the investigation, it became clear that agile artifacts were enriched and tailored with plan-based approaches to handle the companies' compliance, risk, and quality management, which could not be covered with agile in its pure form. Moreover, the paper focused on the artifacts aspect of the teams and not the agile practices tailored to those large-scale settings, which reinforces the need for further research on this theme.

The study from Alqudah and Razali focused on revising six scaling agile methods/frameworks (ALQUDAH; RAZALI, 2016), comprehending their roles and practices, and identifying their differences and similarities. The frameworks chosen were DAD, SAFe, LeSS, LeSS Huge, Spotify, Nexus, and RAGE. Each framework was compared in five criteria, the team size, training and certificate on, methods and practice adopted, technical practices required, and organization type. However, the study mainly focused on investigating the agile frameworks based on their pure use and not on the literature of case studies tailoring them. Due to it, the authors could not evaluate the applicability of the tailoring practices from those frameworks since comparing them was the primary goal. Meanwhile, it reinforces the need for further research in those frameworks to understand better how each one works and how their practices vary according to the agile team use and the market context of the companies.

Dikert *et al.* (DIKERT; PAASIVAARA; LASSENIUS, 2016) conducted an SLR to review how large-scale agile transformations occurred in the industry and described how agile methods and lean software development were adopted and adapted at scale. The study identified challenges and success factors during the agile transformation of large-scale and distributed scenarios. The article points out that 90% of the selected studies for the SLR were experience reports, indicating that academic research is necessary. However, since the focus was mainly on the challenges and success factors of scaling agile, specific agile practices and how they were implemented in GSD settings were not considered. Additionally, the study does not relate the agile practices covered in success factors to the practices of scaling agile frameworks. However, it points to the need to understand better how scaled frameworks are adapted, their challenges and benefits, and how their practices are tailored. Such points indicate the need for further research in large-scale agile distributed environments tailoring agile.

Lous *et al.* (LOUS et al., 2018) presented a case study conducted during one year at Debitoor. This small Danish company develops a single software product with 40 employees distributed across Denmark, Ukraine, and Lithuania. The authors presented a customized process adopted by the company, which emerges from the agile philosophy but embraces the characteristics of distributed development. The study also shows a virtual work environment between two teams in the organization and how the agile practices and tools were carefully selected to support the distributed agile development teams. The paper summarizes a set of practices tailored by the teams since the retrospective was conducted on a 3-weekly cadence, one-on-one through face-to-face walks, growth hacks meetings, and others. Besides the set of tools used by the team to support the development was also presented. Moreover, the collection of tools was assessed through communication, coordination, collaboration, and awareness dimensions; and the communication network among team members was described. The teams' practices were tailored according to the challenges addressed in another study by Lous, which showed 45 challenges in using Scrum in GSD. Ultimately, as pointed out by the authors, they present a prototype for organizing and running distributed projects in a small-scale company in a particular context, which can not be generalized for other contexts. Based on it, we hope to go further and provide a framework that could cover several contexts to describe how to run and organize agile distributed teams with agile tailored practices in different scale dimensions.

Ambler and Lines had updated their original version of the Disciplined Agile Delivery (DAD) toolkit published in 2012 (AMBLER; LINES, 2012). In 2020, the authors released its new version through the book “Choose Your WoW: A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working (WoW)” (LINES; AMBLER, 2019). The DAD became only Disciplined Agile (DA), and the authors focused on fully describing a toolkit allowing the organization to choose its own way of working (WoW) based on a guide that would help individuals and teams make better decisions and tailor their process according to their purpose (LINES; AMBLER, 2019). DA emphasizes a pragmatic, context-based, and enterprise-aware approach to agile and lean delivery (LINES; AMBLER, 2019).

Developing agile software has become popular but is also becoming increasingly complex, involving several individuals and teams from different cultures and organizations with different sizes and processes. In the meantime, some organizations understand that one-size-fits-all solutions may not work for their specific context (TENDEDEZ; FERRARIO; WHITTLE, 2018). Based on such a challenge, Ambler and Lines developed DA beyond the choose your WoW approach (LINES; AMBLER, 2019), which foment, in some way, the agile tailoring approach. In the authors' opinion, regular frameworks, like SAFe (Leffingwell, Dean, 2023), and Scrum (Schwaber, Ken and Sutherland, Jeff, 2022), describe what the individuals, teams, and organizations should do by narrowing the options with a set of best practices, ceremonies, roles, and events. By doing this, the structured frameworks do not respect the context by avoiding the understanding that each individual, team, and company are unique (LINES; AMBLER, 2019). As the authors say, it does not mean that regular agile frameworks do not work, but it may limit the potential of your teams since they solve a problem with some practices but do not foment continuous improvement (LINES; AMBLER, 2019).

Ambler and Lines, in their book, present the DA as a toolkit that provides guided continuous improvement through an agile agnostic tool with explicit options of agile techniques and practices, which allows individuals and teams to experiment and tailor a variety of techniques that will help them choose their fit-for-purpose WoW (LINES; AMBLER, 2019). Through a three-step strategy, the authors suggest that organizations and teams: I) align and start where they are independent of the issues; II) improve their process by trying different techniques following a fit-for-purpose strategy; III) thrive by becoming a learning organization that improves its process continuously by assessing the problems, then applying techniques to solve them, assessing effectiveness, and adopting if it works or abandon it fast (LINES; AMBLER, 2019).

Similar to our goal of developing a framework to describe how and what different market sectors are tailoring agile practices using agile frameworks. Ambler and Lines WoW book presents DA as a guide since it summarizes several agile techniques and practices without reinventing the wheel and encourages organizations to use and tailor them to fit their purposes (LINES; AMBLER, 2019). DA foment guided continuous improvement through a set of techniques from each software engineering discipline, from enterprise architecture to release management, finance, sales, data management, and others (LINES; AMBLER, 2019). Meanwhile, the authors also described in the DA how each technique could be applied, and through a ranking structure of effectiveness, the teams and individuals can assess how effective such a technique is (LINES; AMBLER, 2019). Such effectiveness ranking of the techniques also allows teams, departments, and the whole organization to assess where they are according to the DA guidance (LINES; AMBLER, 2019).

The authors emphasize that there is no one-size-fits-all approach to agile and that teams must carefully consider their unique needs and constraints when selecting and implementing agile practices. By embracing the uniqueness and complexity and providing clear choices with agnostic advice, the book provides guidance on assessing your team's context and offers a range of options for tailoring your WoW based on factors such as team size, organizational structure, and regulatory requirements (LINES; AMBLER, 2019).

The emphasis on choosing your WoW (LINES; AMBLER, 2019) is relevant to our agile tailoring research work since we also consider that the context and specificities of the teams and organizations drive the agile tailoring practices selection process. By studying the DA and the guidance provided in the book, we can better understand how teams and individuals can better customize their work to fit its context. The whole book helped us during the development of the TARGET framework by providing clearance on how organizations can use different techniques and assess their effectiveness. In our goal, agile distributed teams working on large-scale settings could select and tailor agile practices according to their needs and context without being restricted to a set of practices from a single framework.

Edison *et al.* (EDISON; WANG; CONBOY, 2022) conducted a systematic literature review to compare the main scaled agile frameworks and methods. The study focuses on the scaled frameworks SAFe, LeSS, Scrum-at-Scale, DAD, and Spotify model, and also custom-built scaled methods developed by companies to address their needs. The study presented an overview of the research area of large-scale agile methods, presenting the findings of previous systematic reviews and the research gaps left by those studies. Furthermore, the authors compared the scaled agile methods by showing their principles, practices, tools, and metrics. It also shows the original method specifications and the extensions and modifications identified in the empirical papers selected. Finally, challenges and success factors the teams face while scaling agile on a large scale are presented in an organized view divided by each scaled agile method, including the main and custom-built ones. The results of the study point to the different ways of adopting and adapting the practices of several scaling agile frameworks presented in the literature, but it does not consider the relationships among the market sectors of the case studies evaluated and practices modified. Our study aims to structure a framework allowing practitioners and researchers to understand how different market sectors tailor agile practices.

Over the years, the agile tailoring research theme is receiving more attention. In different directions and perspectives, several authors have contributed to this body of knowledge by presenting studies related to tailoring agile roles, adaptation to regular agile methodologies, SLRs related to factors and criteria that led teams to tailor agile, or even reviews of the scaling agile frameworks. However, the set of studies does not focus on agile tailoring within agile distributed teams at large-scale settings, which is the focus of this study. Moreover, the following table describes a summary of the related works 2.7.

Table 1 – Related Works Summary.

Study	Year	Description
(HODA et al., 2010)	2010	Highlights the importance of context during agile adoption and tailoring and how it can impact the team's routine. Since the study was conducted more than ten years ago, we reinforce the need to evaluate new contexts through our SLR to understand how agile is tailored.
(HOSSAIN; BANNERMAN; JEFFERY, 2011)	2011	Hossain and Bannerman conducted a multi-case study on four different GSD projects to evaluate the tailoring of regular agile Scrum practices. However, other agile frameworks were not evaluated.
(BASS, 2013; BASS, 2014; BASS, 2015)	2013	Bass, in his studies, conducted a series of interviews with 46 practitioners from 8 large-scale international companies. During a grounded theory process, he evaluated the tailoring of the Scrum Master and Product Owner roles in those large-scale contexts but without focusing on the agile practices.
(CAMPANELLI; PARREIRAS, 2015)	2014	Campanelli and Parreiras conducted an SLR study to systematically review the existing literature on agile methods tailoring. But the analysis does not focus on distributed agile teams in large-scale environments but on agile teams in general.
(RIZVI; BAGHERI; GASEVIC, 2015)	2015	Rizvi conducted an SLR to identify why companies adopt AGSD, which are the most critical risks and threats in AGSD, and which agile methodologies lead AGSD projects to success. But, the study did not considered scaling agile frameworks.
(ROLLAND et al., 2016)	2015	Rolland present a study that examines the underlying assumptions in existing studies of large-scale software development.
(BASS, 2016a)	2016	The study described tailoring techniques on artifacts related to program governance, product artifacts, release artifacts, Sprint artifacts, and future artifacts categories. However, proper attention was not given to agile practices.
(ALQUDAH; RAZALI, 2016)	2016	The study from Alqudah and Razali focused on revising six scaling agile methods/frameworks, comprehending their roles and practices, and identifying their differences and similarities. However, the authors could not evaluate the applicability of the tailoring practices from those frameworks since comparing them was the primary goal.
(DIKERT; PAASIVAARA; LASSENIUS, 2016)	2016	The authors conducted an SLR to review how large-scale agile transformations occurred in the industry and described how agile methods and lean software development were adopted and adapted at scale. But, the study does not relate the agile practices covered in success factors to the practices of scaling agile frameworks.
(LOUS et al., 2018)	2018	The authors presented a customized process adopted by a Danish company, which emerges from the agile philosophy but embraces the characteristics of distributed development. However, they present a prototype for organizing and running distributed projects in a small-scale company in a particular context, which can not be generalized for other contexts.
(LINES; AMBLER, 2019)	2019	Ambler and Lines, in their book, present the DA as a toolkit that provides guided continuous improvement through an agile agnostic tool with explicit options of agile techniques and practices.
(EDISON; WANG; CONBOY, 2022)	2022	The authors conducted a systematic literature review to compare the main scaled agile frameworks and methods. But, it does not consider the relationships among the market sectors of the case studies evaluated and practices modified.

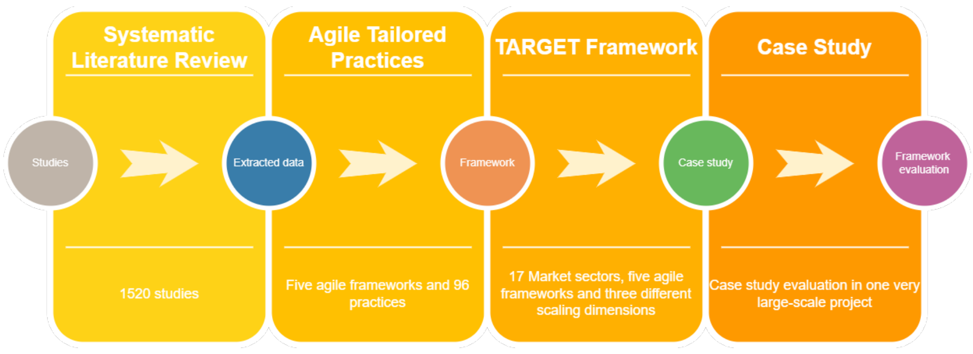
Source: The author (2023)

3 METHODOLOGY

This chapter will explain and describe the methodology used during the three stages of the whole study. First, the Systematic Literature Review (SLR) gathered the literature data that guided the present study. Then, the TARGET framework emerged from the structured data extracted from the SLR. And finally, a case study of a very large-scale project from an IT service provider company to evaluate part of the framework design.

In the following Figure 3, it is possible to see how the three stages have been conducted sequentially in a visual representation. With the input of the SLR studies, we could access a set of agile tailored practices from the literature and then structure it in the TARGET framework for evaluate through a case study.

Figure 1 – Research Methodology Phases


Systematic Literature Review
Agile Tailored Practices
TARGET Framework
Case Study
Studies
Extracted data
Framework
Case study
Framework evaluation
1520 studies
Five agile frameworks and 96 practices
17 Market sectors, five agile frameworks and three different scaling dimensions
Case study evaluation in one very large-scale project

Source: The author (2023)

3.1 SYSTEMATIC LITERATURE REVIEW

A Systematic Literature Review following the guidelines of Kitchenham and Stuart (KITCHENHAM; CHARTERS, 2007) was conducted in this study to evaluate most of the relevant literature regarding agile tailoring practices in large-scale environments used by distributed teams that use agile frameworks. Our goal was not to uncover all recorded tailoring approaches regarding agile practices from every agile and scaling agile framework presented in the software engineering area but to focus on five of the most used in the literature. However, we aim to collect sufficient information about the tailoring of agile practices by distributed teams in large-scale contexts that uses agile and scaling agile frameworks to be the most representative study regarding the adaptation of agile practices.

By doing this SLR, we aim to reveal how the current literature tailors agile practices and methods using different frameworks in large-scale settings. By the end of the SLR, we look forward to being able to answer the following research questions: "What agile practices do the DSD teams that apply agile frameworks in large-scale settings are tailoring?", and "How do DSD teams that apply agile frameworks tailor its agile practices in large-scale settings?".

Three researchers conducted the SLR, the first author and the advisors. During the SLR's phases, the author and the advisor developed the research protocol, and the author executed the search string in the bibliography databases. The search results were exported as BibTeX files, then organized into the StArt software (ZAMBONI et al., 2010), an open-source support tool for SLR research, and then all the articles were evaluated until the full read phase.

Moreover, by building a set of data enough to answer both of those questions during the SLR, we aim to have sufficient information regarding the state of the art of agile tailoring. Within it, it will be possible to develop the basis of a framework to be the leading guide of agile tailoring on large and distributed scenarios involving teams spread in different locations to help practitioners and researchers.

3.1.1 Document selection

We applied automatic search in five large bibliographic databases to identify a set of relevant studies that the research goals should match. The search string aimed to gather the keywords of each research theme by combining the keywords of "distributed software development", "scaling agile", "agile method tailoring", and "large-scale". Based on this structure, the search string can be accessed in the research protocol (See in <https://bit.ly/3EkIpMy>) or in the table 3.1.1. We used this string to search the metadata relating to journals and conference proceedings in IEEE Xplore, ACM Digital Library, SpringerLink, Scopus, and Wiley bibliographic databases.

The search produced 1520 references from 2001 to 2021 (IEEE = 31; ACM = 836; Springer= 191; Wiley = 58; Scopus = 404). The references were limited from 2001 onwards since the Agile Manifesto was published this year. The selection process had three phases: (i) an initial selection of research results that could reasonably satisfy the selection criteria (outlined next) based on a reading of the studies' titles and abstracts; followed by (ii) a selection against these criteria from the initially selected list of studies based on a reading of their introductions and conclusions; (iii) and finally, the studies were fully read and the ones related to the tailoring of agile practices by distributed teams in large-scale contexts that uses agile

Table 2 – Main Search String.

Main Search String
("distributed software development" OR "distributed software engineering" GSE OR GSD OR "distributed teams" OR "global software development" OR "global software engineering" OR "global team" OR "dispersed team" OR "spread team" OR "virtual team" OR "offshore" OR "outsource" OR "DSD" OR "DSE")
AND
("scaling agile" OR "scaled agile framework" OR SAFe OR Spotify OR Scrum@Scale OR Scrum OR Kanban OR Lean OR Nexus OR "large Scale Scrum" OR LeSS OR "agile programme management OR AgilePgM OR XP OR "Extreme Programming" OR "feature driven development" OR fdd)
AND
("agile tailoring" OR "agile software development" OR "agile method tailoring" OR "agile practices tailoring" OR "agile practice tailoring" OR "agile adaptation" OR "agile method adaptation" OR "agile practices adaptation" OR "adapting agile" OR "adapting agile methods" OR "adapting agile practices" OR Agile OR "Agile practice" OR "Agile method")
AND
(large OR scale OR large-scale OR "large-scale development" OR "large-scale agile development" OR "large-scale settings" OR "large-scale environment" OR "large enterprise" OR "large project" OR "large organization" OR "large company")

Source: The author (2023)

framework were selected.

3.1.1.1 *Inclusion/Exclusion criteria*

The following criteria guided the selection of studies. We *included*:

- (IC1) complete, peer-reviewed, published studies;
- (IC2) studies directly related to the research question and its subjects, such as agile tailoring in large-scale distributed settings;
- (IC3) the study is available via the university library services accessible to the authors during the time of the research;
- (IC4) keywords of the research string appear on abstract or author keywords;

We *excluded*:

- (EC1) texts not published in English;
- (EC2) technical content, e.g: editorials, tutorials, keynote speeches, white studies, thesis, dissertations, technical reports, books; and
- (EC3) short studies (<=4 pages);
- (EC4) studies not related to agile tailoring in large-scale distributed settings;
- (EC5) studies related to education matters;

- (EC6) studies that present personal viewpoints or specialists' opinions;
- (EC7) Studies that do not clarify the research area as distributed software development, scaling agile, tailoring agile methods, and large-scale software development.

Each study was checked to ensure there was no duplication or replication. For example, if a given study were published in two different journals with a different order of primary authors, only one study would be included; this would usually be the most comprehensive or recent study. By doing this, it was possible to identify 90 duplicated studies and two replication. After excluding the duplicated ones, the selection phase was executed, and we identified 247 studies for phase 1. During phase 1, the introduction and conclusion of each study were read, and 91 studies were selected for the full read. In phase 2, the total read removed 17 studies. During this phase, studies related to other subjects, such as outsourcing in distributed large-scale settings or general GSD, were removed (CAMARA et al., 2022a), and the final set consisted of 74 studies. Of those 74 studies, each one was evaluated regarding the agile tailoring practices used by the distributed teams. For a better view of the number of studies during each phase and the distribution on the bibliographic databases, see table 3.1.1.1

Table 3 – Studies by engines.

Engine	**Selection**	**Phase 1**	**Phase 2**
IEEE	31	26	15
ACM	836	69	21
Springer	191	61	22
Wiley	58	9	2
Scopus	404	82	14
Total	1520	247	74

Source: The author (2023)

During the selection phase, the author read the title and abstract of all studies, and then the result dataset was discussed with the advisors. Whether two or all the researchers approved an article, it was included in the next phase. However, if a study received only one vote for approval, such a study would be discussed by all the researchers until they reached a consensus. In phase 1, each study of the 247 was accessed and evaluated by the author, and after the evaluation, the selected ones were assessed following the same rules of the selection phase.

In phase 2, 74 studies were fully read by the author. The data were extracted by the researchers in the form of quotes, all the researchers evaluated the dataset of quotes, and any disagreements were discussed until a consensus was reached.

3.1.2 Study Quality

Our study's quality assessment criteria are based on principles and good practices established for driving empirical research in software engineering (DYBA; DINGSOYR; HANSSEN, 2007), briefly summarised as follows. We answered the following questions using: *Yes, No, Partially* (i) is there a clear definition of the study objectives?; (ii) Is there a clear definition of the justifications of the study?; (iii) Is there a theoretical background about the topics of the study?; (iv) Is there a clear definition of the research question (RQ) or the study's hypothesis?; (v) Is there an adequate description of the context in which the research was carried out?; (vi) Is there an adequate description of the data collection methods?; (vii) Is there an adequate description of the sample used and the methods for identifying and recruiting the sample?; (viii) Is there an adequate description of the methods used to analyze data and appropriate methods for ensuring the data analysis was grounded in the data?; (ix) Does the study provide clear answers or justifications about RQ/hypothesis?; (x) Does the study provide clearly stated findings with credible results?; (xi) Does the study provide justified conclusions?; and (xii) does the study discuss validity threats?

3.1.3 Study Evaluation

The evaluation assessment used in the study is based on the method of rigor and industrial relevance evaluation proposed by Ivarsson and Gorschek (IVARSSON; GORSCHEK, 2011). The authors describe an evaluation model of rigor and industrial relevance of technology evaluations in software engineering. The rigor aspects are evaluated in three dimensions: the extent to which (i) context, (ii) study design, and (iii) threats to validity are described. Each aspect is classified on a scale that helps measure and quantify a paper's rigor score. The scale is composed of three possible values: 0 ("weak"), 0.5 ("medium"), and 1 ("high"). The maximum value for rigor is 3.

Conversely, the model considers relevance as a study's potential impact on academia and industry. However, it is also important to consider relevant research topics and real industrial settings to achieve relevancy. Differently from rigor, industrial relevance is evaluated in four aspects that only accept a binary score, 1 for present and 0 for not present. The aspects are (i) the subjects of the study that are described as the people involved in the case, e.g., industry professionals; (ii) the context in which the study was conducted, e.g., industrial settings; (iii) the scale of the applications used in the study, e.g., realistic industrial applications; and (iv) the research method used, some methods like a case study, survey, and action research are supposed to provide real evidence leading to more industrial relevance instead of methods like laboratories experiments, conceptual analysis. The maximum value of relevance is 4.

3.1.4 Data Extraction

Each selected study was fully read to extract information regarding agile tailoring practices of distributed teams that worked in large-scale environments and used agile and scaling agile frameworks.

Data extraction refers to recording all relevant information from the studies required to answer the RQ (WOHLIN et al., 2012). To synthesize the data and ease the management, we conducted some recommended steps by (CRUZES; DYBA, 2011).

We synthesized the data by identifying each tailoring agile practice and how distributed teams apply it in large-scale environments. As we gave each occurrence the same weight, the frequencies presented after the practice title reflect how many papers mention a given practice; frequencies, therefore, reflect the prevalence of a practice and not its potential importance.

Moreover, to have a structured data extraction process and to facilitate the management of the extracted data, we decided to use the strategy of categorizing studies into research type and contribution type facets, as suggested by Petersen et al. (PETERSEN et al., 2008) and Wieringa et al. (WIERINGA et al., 2006).

The following research types facet derived from Wieringa *et al.* (WIERINGA et al., 2006) were considered for evaluation.

- **Solution:** Proposal solution to a problem can be an improvement or a new technique's proposal. The proposed solution must be argued and, when possible, tested and validated;
- **Philosophical:** Proposes a new way of thinking/looking at things. It can be a new conceptual framework, a taxonomy, and a secondary study, such as SLR or SMS;
- **Evaluation:** Evaluation of a problem in practice, or evaluations of a technique implemented in practice, e.g., case studies;

- **Experience:** It describes personal experiences. It can be personal experiences from the authors or industrial experience reports.

Also, all the reviewed studies were classified through the contribution type facets derived from Petersen et al. (PETERSEN et al., 2008).

- **Model:** Representation of an observed reality through concepts;
- **Framework:** A set of practices, methods, and recommendations to be applied;
- **Guideline:** A set of advice, best practices, and success factors grounded in empirical evidence;
- **Lessons learned:** A set of outcomes from case studies findings and results;
- **Advice:** A set of recommendations usually from the author's opinion and not grounded in empirical evidence;
- **Theory:** A construct of cause-effect relationships.

A spreadsheet was used to record the extracted data. Quotes from each study that answered the research question were recorded on separate results forms. We also synthesized the quotes data by identifying themes from the findings reported in each accepted paper following an open-coding process and a constant comparison among the codes (GLASER, 1992). The open-coding activity was executed in the MAXQDA Software [1]. During the synthesis, the codes were grouped into categories, and most of the categories emerged based on the different agile frameworks used by distributed teams in large-scale environments. Categories related to other themes, such as outsourcing of distributed teams in large-scale environments, generated another study (CAMARA et al., 2022a).

3.2 FRAMEWORK

3.2.1 Framework Mapping

The development of the TARGET started during the data extraction of the studies in the second phase of the SLR (see table 3.1.1.1), in which the studies were fully read for a

[1] www.maxqda.com

better understanding of how distributed teams from large-scale environments are using agile and scaling agile frameworks and tailoring the agile practices from those frameworks.

Beyond the extraction of the tailored practices from the studies, its contribution and research types (PETERSEN et al., 2008; WIERINGA et al., 2006), and its rigor and relevance score (IVARSSON; GORSCHEK, 2011). Information regarding the study's characteristics was also extracted to compose the development of the TARGET framework. First, the studies were separated according to their market sector. Second, the taxonomy presented by Dingsoyr *et al.* (DINGSØYR; FÆGRI; ITKONEN, 2014) was used to classify each study regarding the definition of agile scaling according to the number of teams involved, which could be small-scale, large-scale, and very large-scale. Third, each agile framework or methodology used in the studies was registered to help build the framework. Finally, the last level of the framework is formed by tailored agile practices.

Within the information extracted from the final set of studies, it was possible to organize 95 tailored practices and their different use approaches. According to the contexts of the organizations in the cases, we could categorize organizations by their market sectors, scales, agile frameworks, and practices tailored to them. By doing this, the TARGET Framework could become a guide playbook for tailoring agile methods according to the organizations' agile framework, scale size, and, most importantly, its business domain.

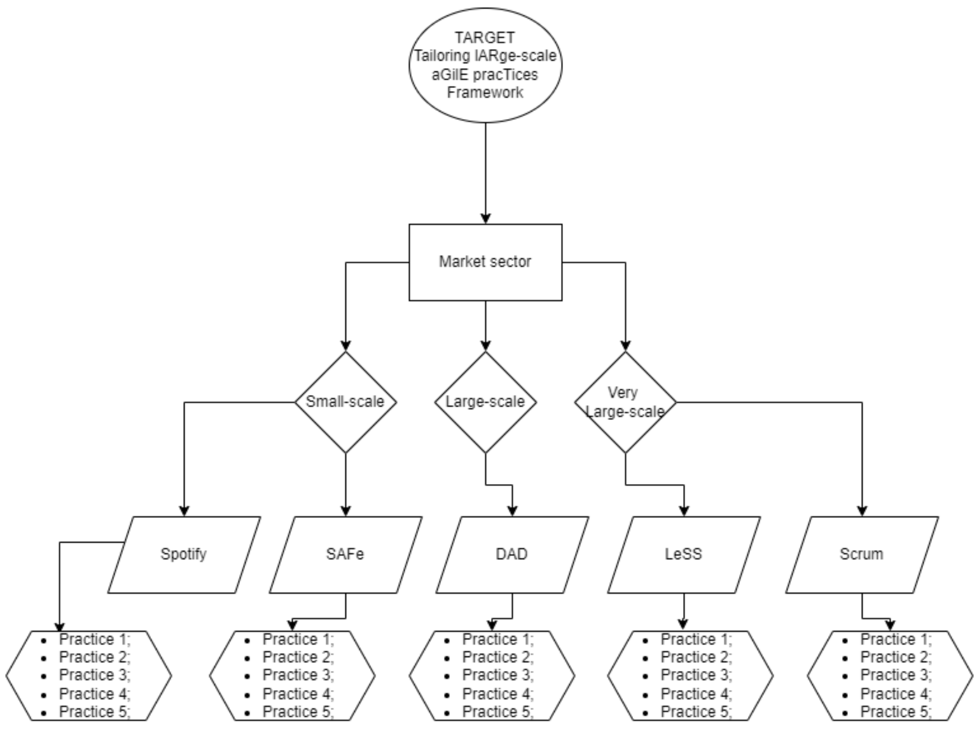
3.2.2 Framework Structure

The TARGET Framework structure aims to provide an overview of how and what different market sectors from the SLR studies with different scaling agile dimensions using different agile frameworks are tailoring agile practices. It is essential to point out that a single study can appear in different domains or scaling terminologies since the sample contains multiple-case studies involving various organizations and studies conducted in different years in the same organization. Moreover, the framework structure must serve as a guide for understanding and inspiring practitioners and researchers by describing how several market sectors with distributed teams in large-scale contexts are developing software.

A specific structure was built for the 17 market sectors identified during the framework construction. Based on the three possible scaling dimensions when identified for the market sector and the agile frameworks used by the studies. Separating the frameworks into 17 individual collections would help researchers and practitioners better understand the tailored practices and asses specific context demands that lead the organization to adapt agile.

The following image describes an example of structure of a market sector in the TARGET Framework (See image TARGET Framework structure example 3.2.2). The framework structure can be viewed as a summary of the tailored agile practices identified in the SLR. Moreover, the description of each of those practices could be consulted in the results of the SLR.

Figure 2 – TARGET Framework structure example.


TARGET Tailoring lARge-scale aGilE pracTices Framework
Market sector
Small-scale
Large-scale
Very Large-scale
Spotify
SAFe
DAD
LeSS
Scrum
Practice 1; Practice 2; Practice 3; Practice 4; Practice 5;

Source: The author (2023)

3.2.3 Framework Evaluation

The framework was built based on state-of-the-art regarding 3.1 agile tailoring with distributed teams in large-scale environments using agile frameworks. Due to it, we believe the evaluation process of the framework must rely on evaluating the tailored practices gathered from the literature on its respective market sectors, scale dimensions covered, and the agile frameworks used. However, it is not feasible to cover the evaluation of the 17 individual collections of the TARGET Framework with real industry cases in this study due to availability, time, and capacity constraints.

Nonetheless, to avoid the absence of an evaluation in the TARGET Framework and compromise the main contribution of this study with such a significant threat to validity. With more studies from the SLR, the market sector of IT service provider companies was evaluated through a real industry case in a Brazilian IT service provider organization. The case study aims to evaluate the presence of the tailored practices identified in the literature by assessing an agile remote distributed team from an IT service provider company that is developing a very large-scale solution for one of its clients. Further, evaluate the similarities of the agile distributed team related to the tailored practices presented in the TARGET Framework regarding IT service provider companies from the literature.

Even though this evaluation step could not generalize the evaluation applicability of the TARGET Framework, we aim, based on the resources of time, capacity, and availability, to cover the most representative market sector from our analysis. In order to reduce the threat to the validity of building a framework without evaluating it in a real industry case scenario.

3.3 CASE STUDY

Case studies are present in areas that aim to study the social and political aspects of individuals, groups, and organizations, such as psychology, sociology, political science, social work, business, and community planning (YIN, 2003). Software engineering is not different. Since individuals, teams, and organizations develop software, the social and political aspects have particular importance in those contexts (RUNESON; HöST, 2009).

The present case study aims to understand the interaction between the studied object, an agile distributed team from an IT Service Provider company, and the tailored practices used by the team in their environment. During the study, the guidelines from Runeson and Höst were followed (RUNESON; HöST, 2009). The study strictly addressed the five major steps from Runeson and Höst (RUNESON; HöST, 2009), which are:

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

Each described step was covered to provide enough credibility and quality for the research case study.

During the SLR, various tailored practices from different scaling agile frameworks were categorized (See section 3.1). Moreover, those practices were organized and helped to build the Tailoring lARge-scale aGilE pracTices Framework - TARGET (See section 3.2), which contained those practices according to the market sector of the companies, its scaling taxonomy, and the agile and scaling framework used by them. After it, it was possible to identify that several IT service provider organizations were seen using diverse tailored practices. Due to it, this case study aims to evaluate the presence of those tailored practices identified in the literature by assessing an agile remote distributed team from an IT service provider company that is developing a very large-scale solution for one of its clients.

Moreover, evaluate the developed framework from the SLR in practice to better understand whether the practices from the most popular sector, IT service providers, are used similarly in an agile distributed team from Brazil working to develop a very large-scale solution. Also, investigate which other practices the teams operate and how different tailoring approaches not seen in the SLR were applied.

Finally, this case study seeks to answer the following research questions: "Which practices from the TARGET framework the agile distributed team from a very large-scale project in an IT service company are using?", and also "How the agile distributed team from a very large-scale project from an IT service company are tailoring agile practices?". By answering such questions, we aim to understand how a real industry case team tailors agile to its context. We also evaluated which approaches presented in the TARGET framework are used by such a team.

3.3.1 Case Study Context

The case study was held in an agile distributed team from an IT service provider company from Brazil with more than 25 years of technology experience and expertise in the different disciplines of Software Engineering. The consultancy firm has headquarters in São Paulo. It provides services on software development, quality, design, software tests, performance tests, observability, security information, LGPD compliance, and others for the entire country and even international clients.

For the case study, an agile distributed team with eight members spread across four out of the five regions of Brazil was selected. The furthest member from the company is separated by 3500 kilometers of distance, and the members never meet each other. Moreover, the Tech leader of a distributed quality team from the same consultancy firm and the client was selected. The tech lead works with more than ten quality analysts across Brazil.

The agile distributed team and the Teach lead from the quality team that represents the unit of analysis are allocated to developing a very large-scale solution for one of the largest educational groups from Brazil and LATAM. The project involves more than twenty teams, 600+ hundred people, and other IT service provider enterprises developing different solution modules. The selected agile distributed team develops one of the modules responsible for the sales control, lead management, and conversion of new candidates in students. In the meantime, the Tech leader is responsible for managing and guiding the quality assurance process of the whole solution by executing functional and developing automated tests for the different systems that compose the very large-scale education solution.

3.3.2 Identification of Unit Analysis

The study happened through teleconferences tools available in the IT service provider company due to the distribution of the team members. The specific agile distributed team chosen to be the unit of analysis is one of the few from the company that works on a very large-scale project involving more than ten teams, several other third-party suppliers, and more than 600+ hundred professionals.

In this very large-scale project, the clients claim to use SAFe Framework (Leffingwell, Dean, 2023) for the development of the solution. However, the teams can work with the agile framework or methodology they want. The specified agile distributed team defines itself as a Scrum (SCHWABER; SUTHERLAND, 2020) team working remotely in solution development.

Based on the self-definition, the case study aims to identify which practices from the IT service provider sector from the TARGET framework the agile distributed team uses and how the team tailors the practices already mapped by the framework. Moreover, which practices not presented in the framework are tailored and used by the agile distributed team. By using this team as a unit of analysis, it is possible to evaluate the tailored practices beyond Scrum (SCHWABER; SUTHERLAND, 2020) as the team specificities itself, but also from SAFe framework (Leffingwell, Dean, 2023) as the client specifies itself.

3.3.3 Data Collection

In this case study, the primary data source was extracted from interviews with members of the agile distributed team. Those interviews were held and recorded through teleconference meetings. However, to ensure a data triangulation process (RUNESON; HöST, 2009), observation of the teams' activities was taken into account by participating in team events, such as dailies, retrospectives, reviews, and similar meetings. Finally, data from the project documentation was also evaluated, such as sprint backlogs, architecture references, and business definitions. Most data extraction methods approach the first degree of data collection techniques as presented by Runeson and Höst in their study (RUNESON; HöST, 2009).

In the interviews, most team members were selected to achieve maximum representation and variation among the unit of analysis. A total of six out of eight people from the agile distributed teams were interviewed. The sample is represented by people with various roles, from quality analysts to frontend and backend developers, a Scrum Master, and a Tech Leader for Quality Assurance. In addition, respondents had experience in the technology area ranging from one and a half years to more than seven years. Finally, all respondents had a degree in technology-related areas, and some had a postgraduate degree. Table 3.3.3 presents the demographic data of the interviewees. To maintain the interviewees' anonymity, they are referenced through codes from P0 to P5.

Table 4 – Demographic data of respondents.

ID	Role	Years of Experience	Time in the project	Age
P0	Quality Analyst	7 years of experience	6,5 months	26
P1	Scrum Master	5 years of experience	2 months	37
P2	Developer	1,7 years of experience	6 months	21
P3	Developer	10 years of experience	6 months	30
P4	Developer	8 years of experience	8 months	30
P5	Tech Leader	10 years of experience	6 months	27

Source: The author (2023)

The interview with the six collaborators followed a semi-structured model that allowed us to gather information about the tailored agile practices used by the team based on the TARGET Framework. Each interview lasted from 1 hour to 1 hour and a half. Of the six interviews, one of them was a pilot. From the pilot interview, it was possible to evaluate the script, make the necessary changes, and go on to the next interviews of the study. It's important to note that the pilot's results were included in the overall study results. Finally, before each interview, consent was previously required, and members were warned about data usage from the recording interview.

The interview script was organized on four topics that can be accessed on <https://bit.ly/3YduQIo>. First, we had the goal to understand the practitioners' context better, then the perception of the team regarding being agile or not, followed by the presentation of the agile-tailored practices that match the context of the team members, and finally, the agile-tailored practices outside the context of the team members. Since the case study members could not have entire knowledge regarding the agile, tailored practices presented to them, each of the practices was previously explained to the interviewees before asking if they were using it or not.

During the observation process, the members did not interact with the researcher. All the meetings were played as usual, with or without the client's presence. This part aimed to better understand the teams in their particular environment and apply data triangulation to verify the identified tailored agile practices. Similarly, the documentation analysis seeks to evaluate the presence of practices in the living documents of the project.

3.3.4 Data Analysis

For the data analysis process, it was used some of the phases present in the Grounded Theory guide from Glaser and Strauss (GLASER; STRAUSS, 2009). However, it is essential to point out that this case study does not cover all the steps necessary to be considered a grounded theory research, only the data analysis steps from a grounded theory were considered.

During this phase, data analysis was started using the open coding technique and the constant comparison method through the transcript data from the interviewees. We sought to synthesize interview data by identifying patterns in the transcripts. During the process, the open coding technique allowed us to look for patterns in the data, while the coding helped define the concepts and descriptions (GLASER, 1992). During the analysis, it was possible to capture several key points of the study and the maximum of tailored practices used by the agile distributed team from the IT service provider.

The main focus of this step was to identify the similarities and differences between the interviewees' speeches and consequently seek meaning in the data extracted from them (MERRIAM, 2009). Then, the consolidated information regarding the tailored agile practices used by the teams was compared to the tailored practices identified in the TARGET Framework. For the entire analysis and coding process, the MaxQDA 2020 software was used and the advisors were responsible for reviewing the coding results.

The open coding process started by selecting transcription texts representing some concept or description of the studied factor. Each new code was compared with existing codes during this stage, whether from the interview itself or others. From then on, every emerging code was constantly compared with the other codes, which allowed the grouping of related codes into categories that imply new agile tailored practices from the team or the already mapped in the TARGET Framework, but with a different approach of use. As the analysis progressed, it was possible to establish relationships between the different categories constructed through the axial coding technique(GLASER; STRAUSS, 2009). However, to better exemplify the described procedures, the category coding process is presented: code freeze.

First, the paragraphs of the transcripts are analyzed, from which it is possible to extract some key points. Such points have a code corresponding to them, where this code represents the key point in a synthesized way, and in addition, a key point can have more than one code. As can be seen below:

Transcript part: "We are planning to use the code freeze technique before regular project releases. We don't use it now, and the developers can publish new code anytime from the delivery pipeline. The client does not care much about the DevOps culture and is trying to grow in it.".

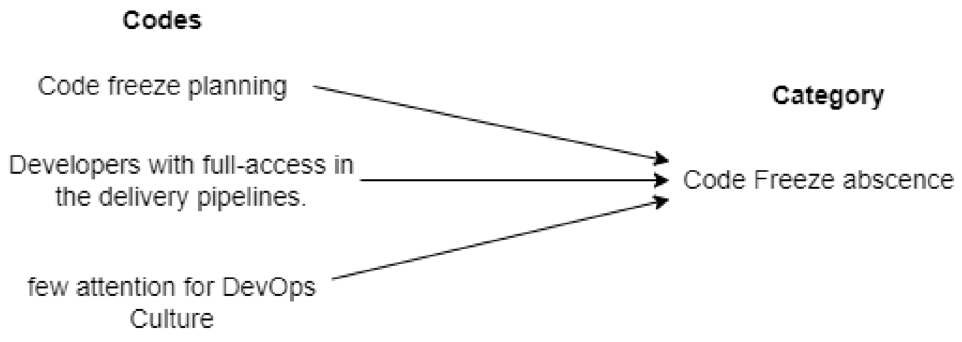
Key Points: “We are planning to use the code freeze technique before regular project releases” and “We don't use it right now, and the developers can publish new code anytime from the delivery pipeline.”.

Code 1 (First part of key point): Code freeze planning.

Code 2 (Second part of key point): Developers with full access to the delivery pipelines.

In the example above, two codes were extracted from P0's interview transcript. Constant comparison with other related emerged codes made it possible to group such codes at a higher level of abstraction. Thus generating the category of code freeze absence (See Figure 3.3.4).

Figure 3 – Code Freeze abscence


Codes
Code freeze planning
Developers with full-access in the delivery pipelines.
few attention for DevOps Culture
Category
Code Freeze abscence

Source: The author (2023)

4 RESULTS

This chapter aims to present the findings of the present study through each of its phases. First, the SLR findings regarding the agile tailored practices used by distributed teams in large-scale environments using agile and scaling agile framework through the analysis of 74 studies from the literature. Then, the TARGET framework breakdown is based on the market sectors from the SLR studies, the agile frameworks used, the scale dimensions, and the adapted agile practices. Finally, the case study results through evaluating the TARGET framework among the most popular market sector.

The findings present in this section will answer the main research questions of this study: "What agile practices do the DSD teams that apply agile and scaling agile frameworks in large-scale settings are tailoring?", and "How do DSD teams that apply agile and scaling agile frameworks tailor its agile practices in large-scale settings?"

4.1 SYSTEMATIC LITERATURE REVIEW

In this section, we present an overview of the studies from the SLR, covering the research and contributions facets of the studies, the research approach adopted, its quality, and the rigor and industrial relevance assessment.

Besides the overview of the studies, the tailoring practices of distributed teams that worked in large-scale environments are presented categorically on its scaling framework or methodology in use.

Since five frameworks and methodologies originated the tailoring practices extracted from the 74 studies evaluated in this study, there are five sessions for each framework grouping the tailoring practices found by teams using them.

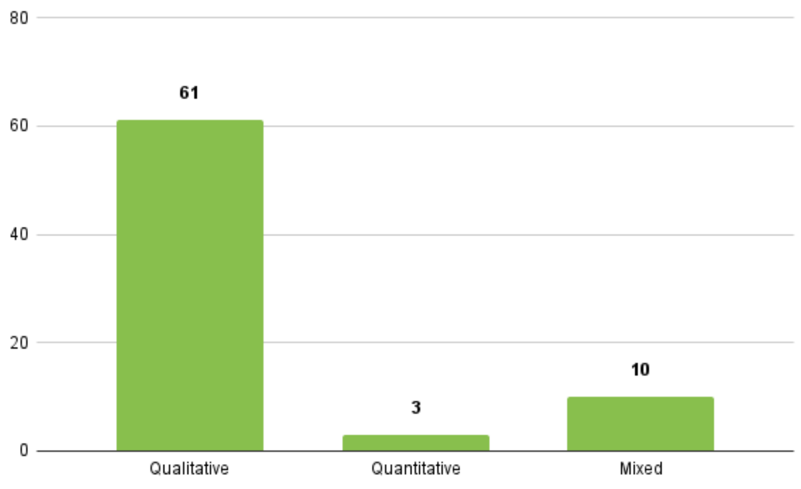
The remainder of this chapt is organized as follows: In section 4.2, we present the overview of the studies from the SLR, describing its methodologies, facets, and other information. Section 4.3 describes the agile tailored practices identified in studies using Spotify. Section 4.4 describes the agile tailored practices identified in studies using SAFe. Section 4.5 describes the agile tailored practices identified in studies using DAD. Section 4.6 describes the agile tailored practices identified in studies using LeSS. Section 4.7 describes the agile tailored practices identified in studies using Scrum. Section 4.8 presents the TARGET Framework structure.

Finally, in section 6, we state some concluding remarks and areas of future research directions.

4.2 OVERVIEW OF THE STUDIES

Of the 74 studies, the majority used a qualitative approach (61 studies), followed by studies that adopted a mixed approach, combining qualitative and quantitative strategies, i.e., a survey instrument combined with interviews (10 studies). Finally, only three studies opted for fully quantitative approaches (See figure 4.2).

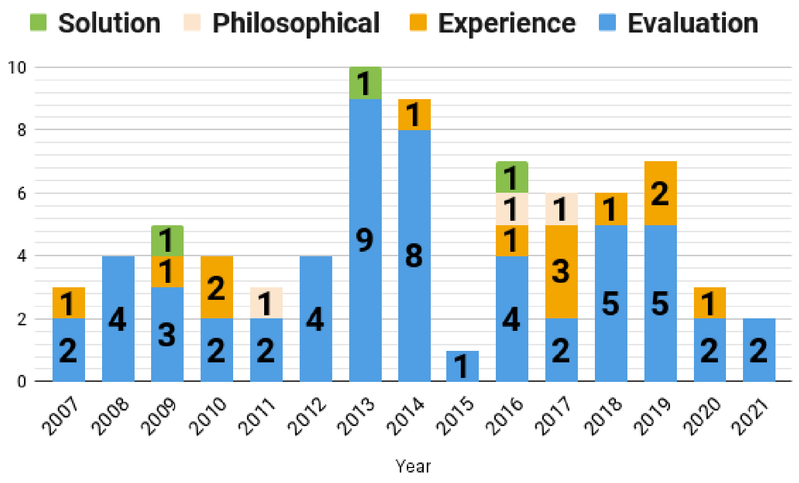
Figure 4 – Research approaches from the studies.



Source: The author (2023)

The 74 studies were also evaluated according to their research type facets. Figure 4.2 shows those facets throughout the years' interval of the studies, from 2007 to 2021. The research type facets are derived from Wieringa *et al.* (WIERINGA et al., 2006), and it aims to classify the papers regarding the research types. At the beginning of the 2000s, no studies about agile tailoring practices in distributed teams at large-scale environments were found. Since agile was born at the beginning of the century, from 2007-2021, the number of studies started to emerge and rise. Most of them were from the evaluation (55 studies), representing 74,32%, experience (13 studies), philosophy (3 studies), and solutions (3 studies). Later, in 2013-2019, we can see more studies, specifically philosophical and solution studies, that indicate a certain maturity of the research field. However, experience and evaluation studies continued to be reported, showing that the research field receives attention from the academy, which regularly researches the area.

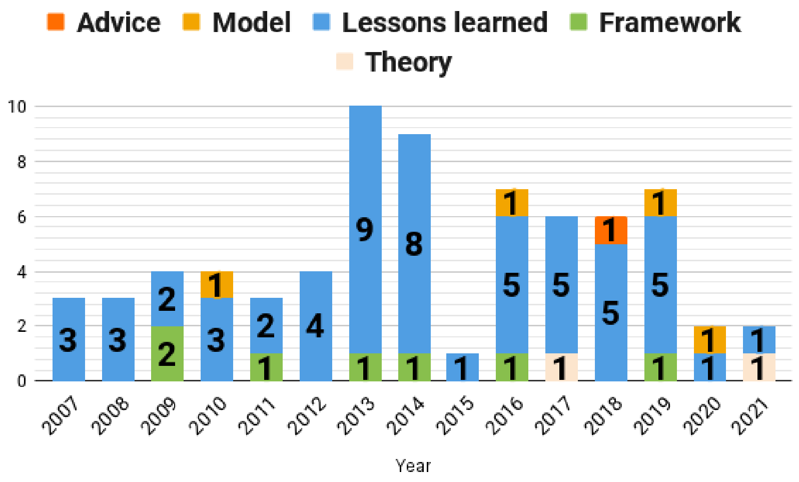
Figure 5 – Research type facets over time.



Source: The author (2023)

The distribution of contribution type facets of the reviewed studies derived from Petersen *et al.* (PETERSEN et al., 2008) and is presented in Figure 4.2. Those facets classify the studies regarding their contribution to the literature. As we can see, the most common contribution types were lessons learned (57 studies), representing 77% of all the studies. Then, the framework (7 studies), followed by the model (4 studies), guideline (3 studies), theory (2 studies), and finally, the advice contribution facet with only one study.

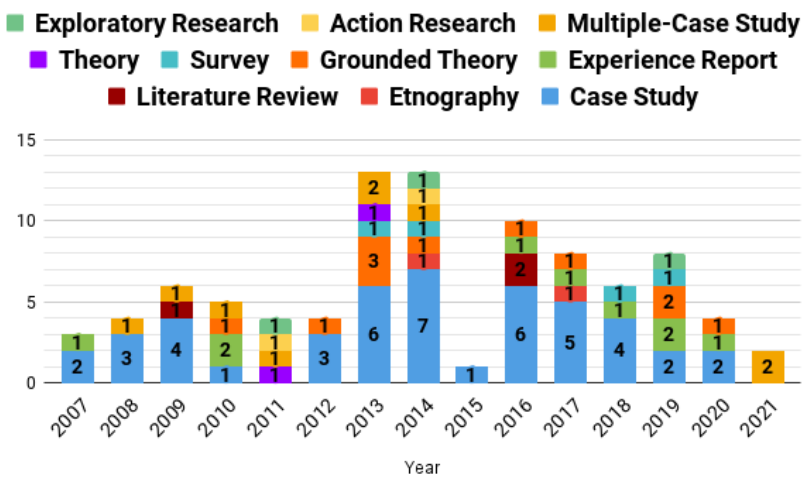
Figure 6 – Contribution type facets over time.



Source: The author (2023)

Most of the research methods used by 74 studies evaluated through this SLR gather case studies and multiple case studies. Beforehand, some studies used more than one methodology, so the number of methods does not correspond to the number of articles. As can be seen in figure 4.2, most studies were classified as case studies (46 studies) and multiple case studies (9 studies), which covers at least one case study in each year from 2007 to 2021. Moreover, grounded theory was also very present in the studies set (11 studies), followed by experience reports (9 studies). Finally, several other research methods were seen, including surveys (4 studies), literature reviews (3 studies), and exploratory research (3 studies) combined with other research methods. The less presented research methods were action research, theory, and ethnography, with two studies each.

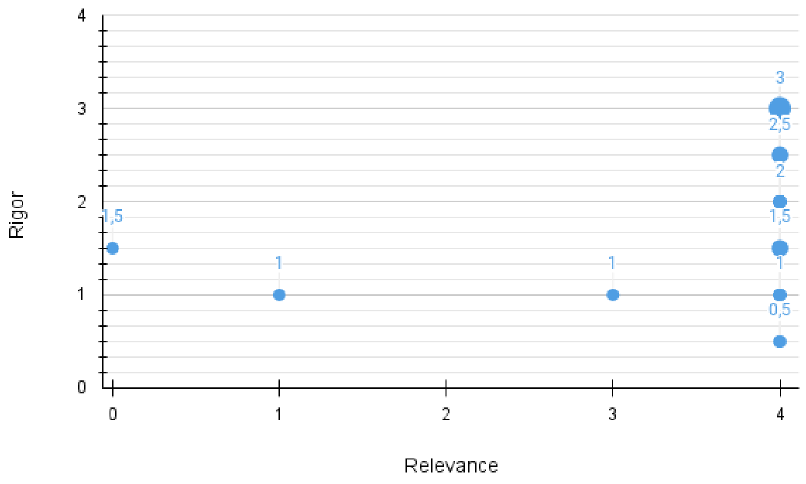
Figure 7 – Research methods over time.



Source: The author (2023)

Regarding the evaluation of the studies, each one was assessed on their aspects regarding rigor and industrial relevance (IVARSSON; GORSCHEK, 2011) (See Section 3.1.3). As shown in Figure 4.2, 20 out of 74 selected studies were classified with the highest score on rigor and relevance, 3 and 4 consecutively, representing almost a quarter of the entire studies, 27%, which indicates a good level of rigor. None of the selected studies reached a 0 on rigor since we look forward to minimally well-structured studies. But, three of them almost got this value, with 0,5 on rigor.

Figure 8 – Rigor and Relevance of the studies.

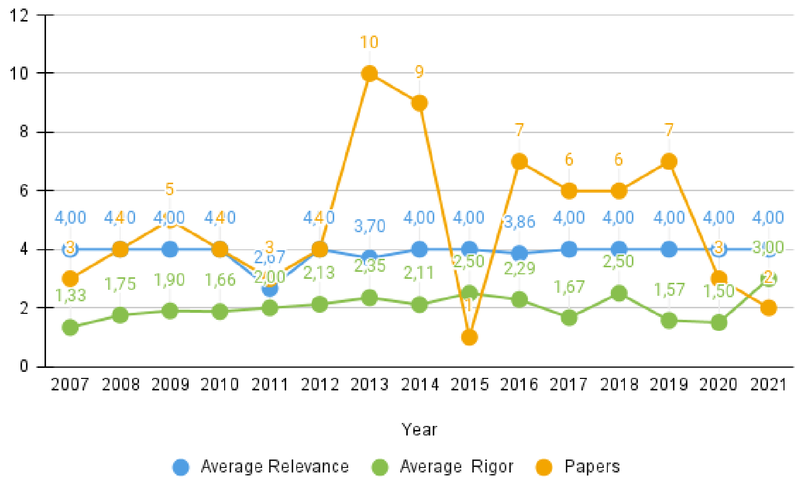


Source: The author (2023)

Conversely, a unique study scored 0 on relevance, which can be considered disappointing from the industrial perspective, but not without proper contribution to academia. However, most papers received quite good scores on rigor and relevance. 71 out of 74 studies were considered with the highest value for industry relevance. Moreover, nine studies show 2.5 on rigor and 4 on relevance, 15 of them 2 on rigor and 4 on relevance, and 17 studies with 3 on rigor and 4 on relevance. Finally, three studies pointed only to 3, 1, and 0 on industry relevance. Three others scored 4 on relevance but only 0.5 in rigor.

Furthermore, the average of rigor and relevance is presented over time in Figure 4.2. Since the selected studies were published between 2007 and 2021, it is possible to see a slight reduction in the number of papers in 2015, which was recovered in the following years. However, the average score of industrial relevance remained stable over the decades, with a slight drop only in 2011, but it also recovered. Conversely, the lowest average rigor score has been seen at the beginning of the first studies published and in recent years of 2019 and 2020, although it improved with the papers published in 2023. Finally, it is important to highlight the growth of studies on tailoring agile in distributed settings at the beginning of the 2010s.

Figure 9 – Average rigor and relevance over time.



Source: The author (2023)

Each study was assessed independently, according to twelve possible quality criteria (see Section 3.1.2). The studies were evaluated on the following scales: <20%, poor; 20%-40%, fair; 40%-60%, average; 60%-80%, good; and >80%, excellent; these are listed in Table 4.2.

Table 5 – Quality Assesment.

	Poor (20%)	Fair (20%-40%)	Average (40%-60%)	Good (60%-80%)	Excellent (80%)
Number of Studies	0	1	10	22	41
Percentage of Papers	0	1,35%	13,51%	29,73%	55,41%

Source: The author (2023)

The quality assessment grades for each criterion (DYBA; DINGSOYR; HANSSEN, 2007) can be accessed in the Analysis sheet in the appendix A.

The following sections present the agile, tailored practices identified in each scaling framework. There is a number next to the title of each subsection. It indicates the frequency of studies that used and adapted the agile practice.

4.3 SPOTIFY TAILORING PRACTICES

4.3.1 Estimation Techniques (2)

Name: estimation techniques.**Goal**: estimation techniques are commonly used by teams to estimate their effort for developing tasks of an iteration. Furthermore, the estimation provides predictability information for companies since it describes when teams will finish their work. Some teams used a scrum technique of story points (SALAMEH; BASS, 2020). **Who**: team members, PO, SM. **How**: Bass and Salameh presented two studies (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020) on a fintech organization that had squads with different missions, going from maintenance, Product-line (PL), and innovation to Proof of Concept (PoC)'s and mini-projects. Due to the different nature of those squads, different estimation techniques were used. The predictability for PL squads, which had low uncertainty in their tasks, was considered beneficial since the product increments could be well planned and estimated. Based on it, the PL teams usually used Lean and Scrumban techniques, such as bucket size and average lead/cycle time (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020). However, squads responsible for the development of new and complex features were dealing with a high degree of uncertainty. Due to this, they combined Lean Startup and Kanban processes to handle the estimation and development (SALAMEH; BASS, 2020). Finally, squads that were working with mini-projects and PoCs were dealing with more uncertainty than any other team. Due to it, those squads employed a tailored scrum process, in which they sometimes used story points to estimate their tasks or even used nothing by just reporting the spent time on each task (SALAMEH; BASS, 2020). According to the authors, those squads considered estimation techniques as a waste activity for innovation teams. Therefore, sacrificing predictability was seen as more important for the sake of innovation and customer value (SALAMEH; BASS, 2020). **Context**: The study conducted by Salameh and Bass (SALAMEH; BASS, 2020) was taken in a fintech organization that used Spotify in 6 different teams with a lot of heterogeneity since they chose different tailored approaches.

4.3.2 Limited Blast Radius Technique (1)

Name: limited blast radius technique.**Goal**: this practice consists of releasing new features of a product to a small portion of the customers instead of the whole users (SALAMEH; BASS, 2019). The technique aims to reduce the risks of incidents by tracking the behavior of the new features with a small group and then rolling it out to others. **Who**: team members. **How**: the study that reports the use of the limited blast radius technique was looking to release software increment on an experimental basis to avoid incidents across the whole user base (SALAMEH; BASS, 2019). Based on it, software was continuously released to a specified number of end-users. Whether it had no problem, the squad may decide to roll out the increment to more end-users until it covers all users (SALAMEH; BASS, 2019). However, when an incident occurred, the squad could roll back the changes and stabilize the environment (SALAMEH; BASS, 2019).

Context: The study conducted by Salameh and Bass (SALAMEH; BASS, 2019) was held in a financial company with mission-critical services for the market. Based on it, the practice was chosen due to critical aspects of the solution that was developed (SALAMEH; BASS, 2019) and to avoid big issues in the whole user chain.

4.3.3 Support/Maintenance Squads (2)

Name: support/maintenance Squads.**Goal**: the purpose of having a support or maintenance squad is to keep specific team members focused on the support of the existing features that could suffer from bugs (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020).**Who**: managers, team members.**How**: in one of the studies conducted by Bass and Salameh (SALAMEH; BASS, 2020), a maintenance squad was built to support the already existed features, although due to the complexity of the software, the squad opted to use Kanban to manage their work. Since Kanban was used, the user stories regarding maintenance were distributed according to the available capacity of resources (SALAMEH; BASS, 2020). Furthermore, the support squad also represented the second level of contact with the customer, helping them with issues investigation and service configuration (SALAMEH; BASS, 2019). **Context**: both studies with support and maintenance squads were held in financial organizations (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020), which require a certain level of support for the customers and stability of the released versions since the tolerance for bugs is low.

4.3.4 Roadmap (2)

Name: Roadmap. **Goal**: the development of a roadmap aims to describe a collection of actions, which will be applied to accomplish both the organization's long- and short- terms goals (SALAMEH; BASS, 2020). **Who**: chapters, team members, managers, PO, and SM. **How**: as presented by Salameh and Bass (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020), the squads had enough autonomy to be aware of what was expected from them, and due to it, the POs' of the squads were responsible for creating the short-term goals that would serve long-term goals of the roadmap. By doing this, the POs' provided the milestones the squads should achieve and a list of actions to fulfill the roadmap (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020). **Context**: in those studies, the roadmap was needed to provide vision and long-term directions for all squads and also organize their job into the desired direction of the organization. Since the squads were dealing with their own tailored practices, the roadmap could combine the milestones of each one in the development of the solutions (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020).

4.3.5 Establish a clear vision (2)

Name: establish a clear vision. **Goal**: establish a clear vision of the solution in development aims to define a scope and a set of specifications, also provide customers, the squads, and stakeholders the direction the solution must take to avoid issues (SALAMEH; BASS, 2019). **Who**: POs and key account managers (KAM). **How**: due to the market volatility, the solution vision should be visited frequently and communicated to everybody involved in the project (SALAMEH; BASS, 2019). The development of a common vision easily creates long- and short-term goals for the squads (SALAMEH; BASS, 2020). POs' and KAMs' should constantly communicate the project vision through regular meetings for the teams to achieve the solution strategy (SALAMEH; BASS, 2019). Despite the customer's intention, the vendor must be aware, guaranteeing that the product development does not deviate from their vision (SALAMEH; BASS, 2020). **Context**: due to the outsourcing nature presented in the studies of Salameh and Bass (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020), the need to report a clear vision of the solution became an important point to ensure the squads into the right direction.

4.3.6 Definition of Done (DoD) (1)

Name: Definition of Done (DoD). **Goal**: the definition of done aims to specify the completeness of a task in a team or squad. It also ensures that a task accomplishes the customer and business needs. **Who**: team, customer, PO, Key Account Manager (KAM). **How**: due to the cross-pollination culture presented in the squads of the case study (SALAMEH; BASS, 2019). The members usually discuss the workflow process together, which leads them to define a standardized definition of done. All squads agreed on the tasks' completeness concept that rules the process flow and satisfies the customers' needs. **Context**: The study conducted by Salameh and Bass (SALAMEH; BASS, 2019) was handled in a financial company that used Spotify to develop a B2B solution for a large-scale mission-critical project.

4.3.7 Postmortem Documentation Process (1)

Name: postmortem document process.**Goal**: postmortem meetings are usually held after a production incident. By involving all team members related to the incident, it aims to produce a list of remediations that need to be taken to prevent the incident from happening again [1]. In one of the studies, the postmortem meeting was tailored to be a postmortem documentation process (SALAMEH; BASS, 2019). **Who**: managers, team members. **How**: Due to the fail-friendly culture introduced by the Spotify framework (LINDERS, 2016), a study reported how they managed the risks that could harm the solution development. To mitigate the future risks of new projects, the organization tailored the postmortem meeting to a postmortem documentation process, in which the team was responsible for listing what was successful or unsuccessful at the end of each project (SALAMEH; BASS, 2019). The postmortem documentation was also filled with customer feedback that was used to improve the product and the development process (SALAMEH; BASS, 2019). **Context**: due to the business domain of the study (SALAMEH; BASS, 2019), a financial organization with B2B mission-critical solutions, the postmortem documentation process was more adequate in the study scenario since it could serve as a documentation reference for a product regulated by the bank sector, and also because this sector does not tolerate failures on the companies reputation.

4.3.8 Measurement Indicators (KPIs) (1)

Name: measurement indicators.**Goal**: measurement indicators are constantly used in the teams to track the progress, code quality, productivity, and performance of the members and/or the whole group (SALAMEH; BASS, 2020). **Who**: managers, team members, PO, and SM using tools.**How**: in one study, the squads were free to tailor their agile process according to their needs (SALAMEH; BASS, 2020). However, the heterogeneity produced by the autonomy harms the measurement of those teams. Since each squad had its own key indicators, different squads could not be compared regarding their quality, velocity, success, and even capacity (SALAMEH; BASS, 2020). After it, the organization realized that they were not able to track the squad's indicators, and must be careful about allowing them to tailor every aspect of their work. **Context**: the study presented by Bass and Salameh (SALAMEH; BASS, 2020) evaluates the heterogeneity of agile tailoring in six different squads. The use of specific indicators in

[1] engineering.atspotify.com/2013/06/incident-management-at-spotify/

each squad has shown the damage caused in the monitoring process of the squads.

4.3.9 Architectural Decision Process (1)

Name: architectural decision process. **Goal**: a large-scale agile project with distributed teams interacting with each other can lead to several architectural decisions every day (SALAMEH; BASS, 2020). However, who should make these decisions? The architectural decision process aims to establish a decision process among the architectural aspects of the solution and elect a architect to be responsible for the design decisions on it. **Who**: architect role. **How**: in the study where six squads constantly interacted during the development of fintech services, the authors reported that the teams lack a process to manage and align architectural decisions among the squads (SALAMEH; BASS, 2020). According to the authors, the absence of such a process impacts the quality of the solution in development. However, its presence may impact the squads' autonomy (SALAMEH; BASS, 2020). However, the architect's role was created to avoid quality problems and the project's complexity. Such a person was responsible for discussing new features with the developers and deciding with the team which architectural change should be made to accommodate the new features (SALAMEH; BASS, 2020). **Context**: in the study presented by Bass and Salameh (SALAMEH; BASS, 2020), the architect's role was required since the chapter leaders were not handling the architectural decisions. Further, the squads were not reaching a consensus.

4.3.10 Knowledge Sharing Process (1)

Name: knowledge sharing process. **Goal**: the knowledge sharing process aims to engage team members in exchanging knowledge regarding common subjects of the project (SALAMEH; BASS, 2020). **Who**: PO, SM, and team members. **How**: the knowledge sharing process in the Spotify framework mainly occurs through guilds formed by people from different tribes, which is called "community of interest" (HENRIK; ANDERS, 2012). However, since the study published by Salameh and Bass (SALAMEH; BASS, 2020) does not have enough scale to have tribes and guilds, on-demand knowledge sharing meetings were arranged to allow squad members to share informal information regarding technical subjects or project domains. Those meetings were arranged through emails and Slack, and anyone interested in the subject could enter (SALAMEH; BASS, 2020). **Context**: at this case, the organization was concerned with the fact the tribes and guilds were inapplicable due to the size of the development program, with less than 100 members (SALAMEH; BASS, 2020). However, despite the regular frequent meetings of communities of interest provided by guilds, the on-demand knowledge sharing meetings were enough to handle the squads' demands without harming their autonomy.

4.3.11 Squad-of-Squads Meeting (1)

Name: Squad-of-Squads meeting. **Goal**: this meeting aims to align all the squads of the project regarding their progress, issues, opportunities, and priorities through a shared Kanban board (SALAMEH; BASS, 2019). **Who**: key stakeholders, key members of the squads. **How**: according to the Spotify Framework, the Scrum of Scrums meeting is usually used for teams to discuss dependencies among their tasks (HENRIK; ANDERS, 2012). However, Spotify squads should not usually hold such meetings since squads are quite independent and don't require this level of synchronization (HENRIK; ANDERS, 2012). Independently of the framework concern, in one case study, the organization considered the Squad-of-Squads meeting necessary to align all squads regarding issues about the behavior of new features released (SALAMEH; BASS, 2019). Therefore, in those meetings, key players of both customer and vendor squads meet up to discuss the progress, identify potential opportunities, and align priorities **Context**: the study conducted by Salameh and Bass describes the development of mission-critical financial services for a B2B market (SALAMEH; BASS, 2019). Since the financial sector requires a high degree of stability in the services, those weekly meetings were a way for the squads from clients and vendors to stay aligned.

4.3.12 Product Owners weekly meeting (1)

Name: Product Owners weekly meeting. **Goal**: weekly meetings with the POs of all squads were held to maintain a shared vision and specifications regarding the solution in development (SALAMEH; BASS, 2019). **Who**: POs'. **How**: the POs' conducted the weekly meeting to align themselves and their squads with the product strategy and the overall roadmap of the organization, reinforce the sense of ownership, and prevent the deviation of the product's main purpose (SALAMEH; BASS, 2019). **Context**: since the study consists of solutions for the financial sector, a wide range of customers can benefit from the introduction of new features, and due to it, those features must always be usable.

4.3.13 Transparency (1)

Name: transparency. **Goal**: build a corporate culture of transparency and mutual respect with the customer. **Who**: the organization. **How**: in the study published by Salameh and Bass, the organization introduced a corporate culture that promotes transparency and mutual respect with the customer (SALAMEH; BASS, 2019). Since the relationship was based on the contract of an outsourced service, the vendor established constant communication regarding what capabilities, time, and resources they could provide to the customer. **Context**: in an outsourcing environment, being transparent to the customer can open new opportunities in future projects (CAMARA et al., 2022b). In this case, vendor transparency and respect fostered the relationship with the client.

4.4 SAFE TAILORING PRACTICES

4.4.1 PI Planning (3)

Name: Program Increment (PI) Planning. **Goal**: PI planning is a regular event from SAFe, in which every team on the Agile Release Train (ART) is aligned to a shared mission and vision (Leffingwell, Dean, 2023). As an output of the PI planning, the teams are committed to the PI objectives, the teams map the dependencies across them, and new features' delivery dates are defined (NOLL et al., 2016). **Who**: every team, Release Train Engineer (RTE). **How**: in the case study presented by Razzak *et al.* (RAZZAK et al., 2018) the program level was more mature than the PPM one, although not enough to conduct the PI planning properly. As mentioned by one of the directors, the project manager supports the PI planning outputs without the participation of any other team (RAZZAK et al., 2018). More than that, practices related to the PI planning, such as the Inspect and Adapt event, were not held. From the case study of Paasivaara (PAASIVAARA, 2017), the teams conducted the with great property. The PI planning event covered two days of schedule. On day one, product managers present business presentations and architecture plans (PAASIVAARA, 2017). On the second day, team-specific planning with some SoS meetings was held, and the teams also had site-specific retrospective meetings regarding the previous PI planning and the current one (PAASIVAARA, 2017). Due to the global distribution of the team, the planning events take place in the main location while real-time Skype voice and video calls are established among the sites (PAASIVAARA, 2017).

Gupta *et al.* (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019) presented an experience report that tailored the PI planning event to a PI Planning workshop. Similar to the approach from the Paasivaara study (PAASIVAARA, 2017), the workshop takes two days. On the first day, the teams focus on refining the backlog current increments aiming to refine the stories to a level that developers and ops find enough information to start the development (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019). After it, the teams discuss the backlog risks for the next two version increments (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019). **Context**: the lack of some important events and ceremonies of SAFecan harm the process. As mentioned by the proper Authors of the framework, the lack of PI planning event configures as a non-application of the SAFe framework (Leffingwell, Dean, 2023). In the study of Razzak *et al.* (RAZZAK et al., 2018), the authors mentioned using some elements of the PI. However, important points were missing, like stakeholders rarely participating in the meetings, not all members attending, and even improvement stories not being properly discussed during the planning. In the Paasivaara case study (PAASIVAARA, 2017), the teams were more committed to regularly adopting the SAFe program level practices. Due to it, both business lines got involved in the PI planning events and could reap good results. In the experience report from Gupta *et al.* (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019), the case takes place in a healthcare project that aims to establish a DevOps approach to support a continuous delivery chain. The authors summarized that the PI planning event was helpful for the projects because since it started, the team released all planned version increments on time and with high customer satisfaction (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019).

4.4.2 External Coaches and Consultants (3)

Name: external coaches and consultants. **Goal**: SAFe is a framework with particular events, roles, and functions. Due to it, the framework recommends a consultant to help the organization during the adoption (PAASIVAARA, 2017). **Who**: coaches and consultants. **How**: in one of the cases from the study of Paasivaara (PAASIVAARA, 2017), a consultant supported the adoption of SAFe through training, workshops for the teams with feedback, and also coaching the RTE in planning activities and arranging the first PI Planning. The team appreciated the consultant since he helped in the managers' coaching and made the teams exercise what they needed to improve (PAASIVAARA, 2017). The other case of the same study had access to an external coach after six months from the beginning of the adoption (PAASIVAARA, 2017). Due to it, the teams faced more problems, and only after some workshops could the team realize which points they needed to improve. This case, in particular, started the SAFe adoption without proper training. Only after facing some problems during the first increments were the product managers, POs, and team members sent to SAFe training (PAASIVAARA, 2017). Further, Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) presented an experience report that a consultant also helped the teams, but especially in fostering the dynamic of Scrum teams that was transitioning to SAFe and considering the current practices, roles, and functions. In Padya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) case, the team member received SAFe training before the ART launch to prepare them better. Lautert *et al.* (LAUTERT; NETO; KOZIEVITCH, 2019) presented a survey study at a large company that uses SAFe. The study aims to understand the preferred type of communication experienced members in agile methodologies would prefer. During the survey, the authors asked whether the team members had taken training on agile and SAFe. The results helped provide training courses to the members that had never received training in the framework (LAUTERT; NETO; KOZIEVITCH, 2019). **Context**: in the Paasivaara case study (PAASIVAARA, 2017), the second case took more advantage of a consultant since the first case just used them six months later. Despite it, the role played an essential role in helping the teams improve the framework practices.Further, in the Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) study, the organization was transitioning from Scrum to SAFe. The transformation took four years, and the presence of a consultant was indispensable since the teams needed to align the SAFe practices from the program level to the team level Scrum practices. Finally, practices regarding estimation, the definition of done, cadence, and synchronization of iterations required the consultants to help to a smoother transition (PANDYA; MANI; PATTANAYAK, 2020). Lautert *et al.* (LAUTERT; NETO; KOZIEVITCH, 2019) has surveyed the company to understand if there is a correlation between years of experience with agile software development teams and prioritized types of communication. However, despite no correlation, the survey helped find which team members required training in the SAFe framework.

4.4.3 Content readiness (2)

Name: content readiness. **Goal**: in SAFe, content readiness is important to ensure clear vision and context for every person in the PI Planning (Leffingwell, Dean, 2023). **Who**: team members responsible for the backlog writing. **How**: to achieve a good level of content readiness before a PI Planning, the teams must involve themselves in a sufficient preparation of the product and architectural backlog (PANDYA; MANI; PATTANAYAK, 2020). In another study (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019), the project manager was responsible for chasing content readiness at the beginning of the project version, which helped team members to understand their common goal. **Context**: in both studies of Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) and Gupta *et al.* (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019), the organizations aim to achieve the content readiness of their backlog to set the teams in the right focus, to avoid unclear expectations from the customer, and also to keep the teams with a vision of the roadmap.

4.4.4 Staff Members for POs' activities (2)

Name: staff members for PO's activities. **Goal**: due to the busy routine POs' were having in such large-scale distributed agile projects. Organizations started to hire additional staff members to execute their regular activities (RAZZAK et al., 2018; BASS, 2015; BEECHAM et al., 2021). **Who**: staff members. **How**: in a case study presented by Razzak *et al.* (RAZZAK et al., 2018) and Beecham *et al.* (BEECHAM et al., 2021) at a small to medium enterprise, the product owners had many responsibilities in their day-to-day routine. The overwhelmed schedule was composed of stakeholder negotiation and prioritization of stories, product management, and acceptance criteria scenarios. Indeed, to reduce the POs' responsibilities and to avoid any deviation from the product roadmap, the company hired staff members to let POs mainly focus on product ownership and the long-term product vision (RAZZAK et al., 2018; BEECHAM et al., 2021). In another study, Bass and Beecham (BASS, 2015) showed how the PO role functions lacked standardization, which led some staff members to hold various job titles and activities. In those cases, the PO teams had onshore staff members to conduct client discussions and offshore staff members to communicate with development teams (BASS, 2015). **Context**: in the case study of Razzak *et al.* (RAZZAK et al., 2018), the authors investigated through surveys the adoption of SAFe in a software company that produces solutions for the optical industry. The authors evaluated the SAFe adoption on three levels, the portfolio, program, and team one, and due to the teams' maturity, some PO functions were still in definitions. In another study, Bass (BASS, 2015) interviewed practitioners from 8 different companies to map how the PO were scaling agile in those large distributed agile projects. Due to the high number of people involved, staff members were necessary to keep things working.

4.4.5 SAFe adoption at Medium Enterprises (1)

Name: SAFe adoption at medium enterprises. **Goal**: medium enterprises can have large-scale distributed projects, although they have to consider which practices, roles, and levels of SAFe adoption is necessary to their environment (RAZZAK et al., 2018). **Who**: the company. **How**: in the case study published by Razzak *et al.* (RAZZAK et al., 2018), the three levels of SAFe: portfolio, program, and team were evaluated regarding the maturity of adoption. During the process, the authors realized that medium enterprises should evaluate which ceremonies, practices, and roles they need while adopting SAFe (RAZZAK et al., 2018). Not everything will be needed, so the results of the self-assessment surveys may help the organizations in this journey.**Context**: in the Razzak *et al.* (RAZZAK et al., 2018) study, the self-assessment survey results were used to specify which practices, roles, and ceremonies would be tailored to fit the medium enterprise company needs.

4.4.6 Project increment workshop (1)

Name: project increment workshop. **Goal**: gather all team members to refine the backlog, and review the processes and metrics of the teams. In other agile methodologies, it is kind of a combination of retrospectives and refinement backlog meetings. **Who**: team members including quality manager, regulatory expert, and operation/back office team. **How**: according to the experience report published by Gupta *et al.* (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019), the teams were conducting project increment workshops which are not well defined in SAFe (Leffingwell, Dean, 2023). As presented in the study, the activity was held for two days rotating the location among the teams' sites: India, USA, and Germany. Independently of the chosen location, members from quality, ops, and development would travel to the destiny. During the workshop, the backlog would be adjusted based on the feedback of a team member from another location, also the quality manager was responsible for helping the teams in refining their process and metrics and enabling a short release cycle with quality (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019). During those workshops, redundant or relevant metrics, checklists, or activities were removed from the process. Finally, the presence of the operation/back office team helped them build knowledge about the development issues and also plan themselves better for the day-to-day activities (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019). **Context**: Gupta *et al.* (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019) presented an experience report at a healthcare project spread across three countries that successfully established a DevOps approach with continuous delivery and short release cycles using agile scrum and SAFe.

4.4.7 Weekly meeting (1)

Name: weekly status meeting. **Goal**: report the project status to the executive management. **Who**: project managers. **How**: in the experience report presented by Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020), the project managers were responsible for reporting their status to the executive management and also deciding internal milestones with them. Also, those project managers needed to present a data-driven report with accurate information capable of anticipating potential questions from the executives and helping to make better decisions. **Context**: the Pandya study *et al.* (PANDYA; MANI; PATTANAYAK, 2020) consisted of an experience report regarding four years of transformation from a Scrum-based organization to the SAFe framework (Leffingwell, Dean, 2023). Those kinds of practices were required to better achieve the business demands during the transformation.

4.4.8 Definition of Done (DoD) (1)

Name: Definition of Done (DoD). **Goal**: define a common definition of done to the teams, and projects aim to establish the completeness of user stories regarding the business value and quality of its delivery. **Who**: managers, team members. **How**: Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) presented a study in which an organization was transitioning from Scrum to SAFe. During the alignment of the program level practices, the organization required changes and adaptation. Therefore, the teams needed to work with a common definition of done with the help of an external consultant. **Context**: since the Pandya study *et al.* describes the transformation from a Scrum-based organization to the SAFe framework (Leffingwell, Dean, 2023), the need to tailor some practices from the Scrum team level brings the attention of the program level presented in SAFe.

4.4.9 Program and Team Boards (1)

Name: Program and Team Boards. **Goal**: use kanban boards to track the PI execution progress during the time, and also the project progress into the teams. **Who**: team members, SM and POs. **How**: according to the experience report published by Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) the digital boards were used during the remote PI planning events and also to track the progress of the execution. After some PIs, the dashboards helped the organization to understand the teams' predictability and throughput. Further, similar project metrics were used in those digital program boards. Finally, some teams used physical dashboards to track their progress in a PI. **Context**: due to the nature of using Scrum and transitioning to SAFe in a distributed environment, led the teams to adopt kanban boards during remote events and combine the use with physical dashboards (PANDYA; MANI; PATTANAYAK, 2020).

4.4.10 Scrum of Scrums (SoS) (1)

Name: Scrum of Scrums (SoS). **Goal**: according to SAFe (Leffingwell, Dean, 2023), the SoS aims to coordinate dependencies across different scrum teams providing visibility through the progress and impediments of the ART. **Who**: RTE, teams. **How**: the case study published by Paasivaara (PAASIVAARA, 2017) at Comptel presented how a business line applied the SoS meetings. First, the SoS meetings were part of the 2-day PI planning event. In the first moment, the teams discussed architecture plans and business vision. Then, one or two SoS meetings to check the status and coordinate the planning were held (PAASIVAARA, 2017). The RTE was the person responsible for coordinating and arranging regular SoS meetings. **Context**: Paasivaara, in this study, presents a case study at Comptel (PAASIVAARA, 2017). The study evaluated the SAFe adoption difference between the two business lines. Despite the comparison, both business lines establish this kind of SoS meeting as part of the PI planning.

4.4.11 Automated tests (1)

Name: automated tests. **Goal**: test automation can be used with continuous delivery to provide quick releases while guaranteeing the quality of components, integrations, interfaces, and acceptance tests (Leffingwell, Dean, 2023). **Who**: quality analysts. **How**: in a multiple case study presented by Beecham *et al.* (BEECHAM et al., 2021), a very large-scale project used automated tests to shorten its release cycle (BEECHAM et al., 2021). The test automation reduced the regressive tests from 6 weeks to 2 weeks, preventing the team from spending great efforts on environment configuration and manual testing. Further, the test automation tasks were the full responsibility of QAs, achieving almost 100% cover. **Context**: Beecham *et al.* (BEECHAM et al., 2021) presented a study to evaluate to what degree scaling frameworks address global software development risks. In one case, the authors evaluated whether SAFe practices could eliminate or mitigate most GSD risks of a large-scale project at an optical industry company.

4.4.12 Feature team (1)

Name: Feature Team. **Goal**: according to (Leffingwell, Dean, 2023), a feature team is organized around user-centered functionalities, in which the main focus is to maintain and enhance the core product. **Who**: team members. **How**: in one of the cases presented by Beecham *et al.* (BEECHAM et al., 2021), the feature team was not able to work on the improvement of the solution. However, they were putting more effort into bug fixes and issues raised by the customer. Due to it, the product roadmap was not followed, which caused delays in important features that needed to be released. **Context**: in this case of Beecham *et al.* (BEECHAM et al., 2021), the deviation of the feature teams to a support team occurred due to the lack of communication of emerging requirements to the product owners that were responsible for the product vision and roadmap. With proper communication, the product owners would be able to drive the solution to a balanced number of fixes, and new features during the interactions (BEECHAM et al., 2021). Further, the company, in this case, moved part-time product owners to full-time personnel that was able to focus solely on the PO role to reduce the impact of emerging requirements.

4.4.13 Single product backlog (1)

Name: single product backlog. **Goal**: in SAFe, teams mainly work with program and solution backlog, which respectively deal with upcoming features that deliver business value, and upcoming capabilities and enablers to build the architectural runaway (Leffingwell, Dean, 2023). However, a single product backlog comprises features, capabilities, and enablers in the experience report by Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020). **Who**: team members, product owner. **How**: to achieve predictability through a high throughput of a product backlog that concentrates new features, bug fixes, and other items, the team organized themselves through upfront planning and prioritization based on a single product backlog for all the teams (PANDYA; MANI; PATTANAYAK, 2020). By doing it, the teams achieved a yearly release schedule, which helped acquire high predictability. **Context**: Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) presented a experience report on software development team in Idia. The report covers four years of transformation from a Scrum-based organization to the SAFe framework.

4.4.14 Measurement Indicators (KPIs) (1)

Name: measurement indicators. **Goal**: keeping track of the solution requires measurement indicators (Leffingwell, Dean, 2023) that also help to track how the value stream is performing against its forecasted outcomes. **Who**: team members, Agile Train Engineer, managers, and others. **How**: in the study published by Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020), Key Performance Indicator (KPI)'s were used to measure different aspects of the solution, such as the project releases, product quality, and team performance. In the field of releases, KPIs were used to track the scope progress, schedule, and cost through release and team burndown, features throughput, project milestones, and cost variance. Regarding the solution quality, the number of defects at the team level, the feature "done-mess", static code analysis, and non-functional requirements trend were the KPIs used. Finally, the organization used the team's predictability, velocity, cycle lead time, and defect fix rate to measure performance. **Context**: the experience report presented by Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) describes a four-year transformation of a Scrum-based organization to SAFe. The nature of the journey required plenty of KPIs to keep track of the changes and to provide visibility that the company is going in the right direction.

4.4.15 Keep stakeholders close (1)

Name: keep stakeholders close. **Goal**: keeping track of key stakeholders' expectations is vital to the success of the solution (PANDYA; MANI; PATTANAYAK, 2020). Teams must meet regularly with stakeholders to better understand their views and to acquire feedback. **Who**: team members. **How**: in an experience report, teams seeking feedback from key stakeholders establish regular meetings with them to identify gaps and gain trust (PANDYA; MANI; PATTANAYAK, 2020). However, only the onshore team had access to it. Further, the teams worked on a stakeholder map by identifying the level of influence and interest of each one and established an engagement plan based on it (PANDYA; MANI; PATTANAYAK, 2020). Each key stakeholder was interviewed and mapped to a four-quadrant structure that considered their attitude and expectations (PANDYA; MANI; PATTANAYAK, 2020). **Context**: in this study (PANDYA; MANI; PATTANAYAK, 2020), the stakeholders played an important role in the solution roadmap. Depending on the project phase, new interviewers were required to deal with those stakeholders' changed expectations. Due to it, keeping track of them was a matter of driving the solution to success.

4.4.16 Instructor-led training (1)

Name: instructor-led training. **Goal**: training team members in technical skills, process, and tools operation during the development process (PANDYA; MANI; PATTANAYAK, 2020). **Who**: mentor. **How**: in the study presented by Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020), the training sessions were made face-to-face on-site by a mentor. The travel costs were budgeted in the project, and the impact of the training during the ongoing projects was carefully addressed in the planning meetings (PANDYA; MANI; PATTANAYAK, 2020) through the reduction of available effort. **Context**: in this experience report (PANDYA; MANI; PATTANAYAK, 2020), offshore teams were working with onshore teams. Keeping both teams in the same process, skills, and tools perspective required face-to-face training sessions by a mentor.

4.4.17 Strategic Themes (1)

Name: strategic themes. **Goal**: strategic themes play an important role in the portfolio decision-making process (Leffingwell, Dean, 2023). It provides a business context that helps connect the portfolio to the enterprise's strategy (Leffingwell, Dean, 2023). **Who**: Program and Portfolio Management (PPM), directors, board. **How**: according to SAFe, Objectives and Key Results (OKR)s are used during the development of strategic themes or even sentences that can influence everyone in the solution development (Leffingwell, Dean, 2023). However, in the Razzak *et al.* (RAZZAK et al., 2018) study, the director of development reported that the organization often used strategic themes to connect the Portfolio vision to the business strategy in an informal way. The PPM team was known as a team that did not foster estimation techniques and planning, even when one of the strategic themes aimed to embrace agile in the organization (RAZZAK et al., 2018). **Context**: in the study (RAZZAK et al., 2018), the assessment at the portfolio level has shown that the company needed to improve the applicability of some practices regarding strategy, investments, funding, value streams, and budget. This result is mostly due to how the PPM team led the activities.

4.4.18 Epic Stories (1)

Name: epic stories. **Goal**: an epic in SAFe is defined as a substantial package of information that requires analysis, using a definition of an Minimum Viable Product (MVP), also describes a significant initiative of development that covers different values streams, program increments and financial approval before implementation (Leffingwell, Dean, 2023). **Who**: PPM team. **How**: the CTO from the company studied at Razzak *et al.* (RAZZAK et al., 2018) has stated the PPM team members were not working with an epic-based at the portfolio level. The PPM area worked as a regular team, focusing on projects, deliverables, and contracts that generated epics used inside the team's environment (RAZZAK et al., 2018). **Context**: in the study (RAZZAK et al., 2018), the PPM maturity level was low, which caused the lack of important practices such as epics in the portfolio management area.

4.4.19 Sprints (1)

Name: sprints. **Goal**: sprints are cycle time iterations in which teams compromise themselves to deliver specific software increments for the solution of an ART (Leffingwell, Dean, 2023). **Who**: teams. **How**: in the same case study presented by Razzak *et al.* (RAZZAK et al., 2018), the teams did not respect some pre-activities and rules. One of the teams reported that managers added almost all open tickets to the sprint during planning. Due to it, the team put the tickets without a proper estimation, which hindered the team from achieving the goal of the iteration (RAZZAK et al., 2018). One of the teams reported missing information during estimation and a lack of time. A developer reported that managers introduced critical tasks into current sprints instead of respecting the rule of adding them in the next iteration. **Context**: in general, the case study of Razzak *et al.* (RAZZAK et al., 2018) has shown some maturity in the sprint activities. However, some specific rules and points of sprints still need to be covered to avoid problems regarding sprint health.

4.4.20 Retrospectives (1)

Name: retrospectives. **Goal**: a retrospective is an event where team members discuss the results of the previous interaction, review their practices, identify ways to improve, and define some actions for the next iterations (Leffingwell, Dean, 2023). **Who**: team members. **How**: according to SAFe (Leffingwell, Dean, 2023), the retrospective must be conducted after the end of each iteration and must be time-boxed for an hour or less. However, the teams rarely hold retrospectives after each sprint at the team level of the case study conducted by Razzak *et al.* (RAZZAK et al., 2018). One of the developers reported that only one retrospective during the last two years was done after a release and not a simple iteration (RAZZAK et al., 2018). **Context**: the lack of such an important ceremony in a single agile team can greatly impact their performance. The absence of a retrospective can reduce the improvement capacity of the team since the members do not discuss ways to improve or avoid negative behaviors. Razzak *et al.* (RAZZAK et al., 2018) did not discuss further implications regarding retrospectives.

4.4.21 User stories (1)

Name: user stories. **Goal**: similar to SCRUM (Schwaber, Ken and Sutherland, Jeff, 2022), stories in SAFe are short descriptions of a feature that needs to be implemented (Leffingwell, Dean, 2023). It also may describe the functionality in the user's language. **Who**: product owner. **How**: the case study presented by Razzak *et al.* (RAZZAK et al., 2018) has shown a different scenario of development due to its nature. A developer reported that the project has little development of user stories since they often work with prioritization and negotiation directly with the client (RAZZAK et al., 2018). To better understand what they need to develop, they need to understand the big documents and specifications of the client. At the same time, the PO was responsible for prioritization and negotiation with the customer (RAZZAK et al., 2018). Due to it, the PO job consisted of conversations to translate the specifications to the team without focusing on user story development. **Context**: one of the study's project managers of the study (RAZZAK et al., 2018) stated that the nature of the contract with the customer was why the teams don't work with user stories. Indeed, the optical industry customer had specific deliverables that were part of the contract, which did not require refinement for user stories.

4.4.22 ART for Business Lines (1)

Name: ART for business lines. **Goal**: The Agile Release Train ART is a combination of agile teams that works together with stakeholders to develop, deliver incrementally, and also can operate one or more solutions in a value stream (Leffingwell, Dean, 2023). **Who**: teams, stakeholders. **How**: Passiavaara, in her case study (PAASIVAARA, 2017) has shown how a company has divided two different business lines in ARTs. Each business line had one agile release train, although two platform teams were serving both business lines (PAASIVAARA, 2017). Due to it, the teams participated in both PI planning events since they compromised themselves in delivering functionalities for both business lines. **Context**: in the case study, Paassivaara (PAASIVAARA, 2017) was evaluating the adoption of SAFe in two different business lines of Comptel, a huge telecommunication company with employees around the world. In this case, each business line had its ART. However, different teams have been working for different business lines, which led them to participate in different PIs and to get involved with many sectors of the organization.

4.4.23 Change Agent (1)

Name: change agents. **Goal**: change agent is a definition of a role responsible for supporting the teams during the change and transition to the SAFe framework (EBERT; PAASIVAARA, 2017). They are also responsible for giving training and workshops and contributing to tailoring the practices. Those change agents can be the RTE, coaches, and managers. **Who**: change agents. **How**: in Paasivaara's *et al.* (EBERT; PAASIVAARA, 2017; PAASIVAARA, 2017) studies, the authors had presented the function of change agents. In the first case of (PAASIVAARA, 2017), the change agent was the RTE that worked part-time. This approach led the RTE not to give proper attention as he would like to for the role that deserved more visible personnel. However, in the second case (PAASIVAARA, 2017) and in the other study (EBERT; PAASIVAARA, 2017) with the output from the first transition, the change agents were R&D, coaches, and RTE. They were working on pushing the change through workshops, training, and exercises through a full-time journey to achieve continuous improvement(EBERT; PAASIVAARA, 2017; PAASIVAARA, 2017). **Context**: as we have seen, in the Paasivaara *et al.* (EBERT; PAASIVAARA, 2017; PAASIVAARA, 2017) studies, the second case took more advantage by having all the information regarding the adoption of the first case. Due to it, the second case better addressed the issues while involving change agents to achieve success.

4.4.24 Release Train Engineer (1)

Name: Release Train Engineer (RTE). **Goal**: according to SAFe, the RTE is responsible for facilitating the major events of the ART (Leffingwell, Dean, 2023). They are also responsible for assisting the process and coaching the team to deliver value (Leffingwell, Dean, 2023). **Who**: release train engineer. **How**: SAFe does not specify the work time of an RTE, although due to its responsibilities, it may be a full-time role (Leffingwell, Dean, 2023). However, in the first case from Paasivaara's study (PAASIVAARA, 2017), the organization chose to work with a part-time RTE. As a part-time in the role, he put some efforts that were appreciated by the teams, although it was not enough for the demand (PAASIVAARA, 2017). The teams realized that the RTE could not push the recognized improvement items forward, and the feeling was that nobody was systematically leading the improvements. However, the second case learned from it, and the RTE worked full-time, which was considered one of the success factors of the SAFe adoption (PAASIVAARA, 2017). Moreover, he was responsible for leading the PI Planning, the SoS meetings, and taking care of the improvement items continuously (PAASIVAARA, 2017). The PI planning in the first case was chaotic due to a lack of preparation from the RTE, although in the second case, the teams prepared themself better (PAASIVAARA, 2017). **Context**: the second case from Paasiavaara's study (PAASIVAARA, 2017) bet on the full-time participation of an RTE, which led to benefits. Since the company was passing through SAFe adoption, the RTE was not supposed to work on a part-time model, and the choice from case 1 was able to show the issues it caused. Finally, the first adoption also served as a driver of how to do things correctly in the second case.

4.5 DAD TAILORING PRACTICES

4.5.1 Risk Mitigation (1)

Name: risk mitigation. **Goal**: Disciplined Agile Delivery (DAD) adopts an approach called risk-value driven lifecycle (AMBLER; LINES, 2012). In this approach, the risk-related features are considered high-priority items, not high-value ones. By doing this, the DAD framework addresses the risks related to delivery as soon as possible by using practices that ensure potentially consumable solutions every iteration (AMBLER; LINES, 2012). **Who**: the teams and the organizations. **How**: Beecham *et al.* (BEECHAM et al., 2021) study has evaluated how two large organizations that use DAD (AMBLER; LINES, 2012) and SAFe (Leffingwell, Dean, 2023) frameworks practices addressed the GSD risks catalog developed by the authors. According to the study, the original DAD practices addressed the GSD risks identified in the literature (BEECHAM et al., 2021). However, only half of the risks from the catalog were observed at the company using DAD. At the same time, almost all the practices identified were implemented except for five that could address the risks from the GSD risk catalog. **Context**: in Beecham *et al.* (BEECHAM et al., 2021) study, the case company was using DAD for the last five years while developing asset management software with ten teams spread across Australia, the USA, and India. The DAD application was so mature that the organization could address every GSD risk identified by the authors just by using the already known risk- and value-driven approach.

4.5.2 Spikes (1)

Name: spike. **Goal**: spikes validate one or more technical approaches before deciding on one design over the others (AMBLER; LINES, 2012). It is more specifically related to architecture, and the team uses it to explore architectural choices before committing to an inaccurate approach (AMBLER; LINES, 2012). **Who**: developers, architectures, and development team members. **How**: in Beecham *et al.* (BEECHAM et al., 2021) study, the company case that used DAD reported the use of spikes to mitigate some risks. First, the teams were developing automated tests in part of their components while unfamiliar with this. Due to it, the use of spikes was helpful to prototype an architecture model that could support the automation and also to develop an early understanding of the technical aspects during the development lifecycle (BEECHAM et al., 2021).Further, conflict requirements were standard during the inception phase. Due to it, the team started to analyze risks and develop a spike on those requirement conflicts before the user stories were set to verify the best technical approach to be adopted (BEECHAM et al., 2021). **Context**: the use of DAD in this company case represents a very mature level of implementation of its practices (BEECHAM et al., 2021). The use of Spikes was taken beyond architectural prototyping, establishing the development of stories and keeping the business goals.

4.5.3 Definition of Done (DoD) (1)

Name: definition of done (DoD). **Goal**: DAD framework does not specify a unique definition of the done concept. However, it discusses the definition of done from the lean community, which describes that the work item is only done when it has been delivered into the user's hands, and they like it (AMBLER; LINES, 2012). In the DAD vision, the DoD can mean that the solution is only done when the users successfully consume it. However, according to the framework, the concept may evolve during the agile adoption (AMBLER; LINES, 2012). **Who**: the team. **How**: Lal and Clear presented a large case study (LAL; CLEAR, 2018) from a company transitioning to DAD. During the process, the very large-scale environment required ten teams to work on different pieces of the solution and to deliver it with high quality. In this case, the DoD practice aimed to ensure zero bug interaction in the short cycle for high quality by being a governance practice combined with unit and acceptance tests, continuous delivery, and pair programming (LAL; CLEAR, 2018). In the study, the DoD focuses on ensuring the quality of the deliverables, but not specifically the consumption of them by the end users. **Context**: Lal and Clear conducted a long case study for more than 15 years in a major Australia-based multisite global software vendor (LAL; CLEAR, 2018). The study consists of a large transition of 10 global distributed teams spread across the USA, Australia, and India from a structured development process, Rational Unified Process (RUP), through a Hybrid-Agile method, and finally to a Scaled Agile approach using Disciplined Agile delivery. By the time the study was presented, the authors had evaluated how the practices, roles, and responsibilities changed in each phase due to the dynamics of a more agile approach (LAL; CLEAR, 2018). Using DAD, the company started to include DoD and other agile practices.

4.5.4 Daily Tactical Huddle (1)

Name: daily tactical huddle. **Goal**: the dailies from DAD do not differ much from regular dailies of Scrum (Schwaber, Ken and Sutherland, Jeff, 2022). However, the DAD has a daily coordination meeting, in which team members organize themselves to decide what they will do in the present work day (AMBLER; LINES, 2012). DAD suggests adopters start with a daily coordination meeting similar to Scrum and then evolve to a Kaban-like coordination meeting, focusing on the work itself instead of the individual (AMBLER; LINES, 2012). **Who**: the team. **How**: Beecham *et al.* (BEECHAM et al., 2021) has shown that the case using DAD implement what they called a daily tactical huddle. The tactical huddle included only the leadership roles, which included the architect, tech lead, and PO, like a daily coordination meeting of leaders. Such daily was required since the organization worked in the program and project division. The regular dailies were concerned with local teams at the project level and were conducted in a co-located manner. On the other side, daily tactical huddles concentrated the leadership to discuss the program's progress on a global level (BEECHAM et al., 2021). **Context**: the case company using DAD on Beecham *et al.* (BEECHAM et al., 2021) works with several projects that together deliver the program. Due to it, the program activities were held globally with leadership with a more accurate holistic view of the projects. Meanwhile, every co-located team across Australia, the USA, and India was concerned with their regular Scrum daily discussing day-to-day work items.

4.5.5 User Stories (1)

Name: user stories. **Goal**: similar to other agile methodologies and frameworks, in DAD, the user stories are work items that describe a requirement that describes a valuable functionality that needs to be implemented (AMBLER; LINES, 2012). However, DAD does not prescribe specifically the user stories. It just recommends a usage-driven approach with a requirement artifact focused on usage, like user stories, usage scenarios, or even use cases (AMBLER; LINES, 2012). **Who**: the team, PO, and analysts. **How**: Beecham *et al.* (BEECHAM et al., 2021) study does not focus on how user stories were developed in both cases. However, one of the team members from the company that used DAD has stated that the continuous focus of the organization in developing user stories was the underline problem of conflict requirements and unclarified requirements (BEECHAM et al., 2021). In his concern, the user stories are important to keep the workflow, although the most interesting insights happen in the day-to-day adjustments (BEECHAM et al., 2021). Finally, he suggests adopting a more lively cycle of adjustments rather than focusing on just design stories for design's sake. **Context**: the case company using DAD on Beecham *et al.* (BEECHAM et al., 2021) study develops asset management software for enterprise demands. The focus on requirement stability is understandable, although the customers' needs are fluid. Due to it, it is important to consider spending some effort on good requirements elicitation to avoid further issues.

4.5.6 Integration and Unit Testing (1)

Name: integration and unit testing. **Goal**: DAD frameworks encourage the use of integration and unit testing (AMBLER; LINES, 2012). Unit testing is part of the TDD approach that development teams must adopt to write just enough unit tests to pass the functionality and validate the expected results (AMBLER; LINES, 2012). DAD also suggests having a suite of unit tests that can be run automatically during the regression test phase, focusing on having 80% code coverage (AMBLER; LINES, 2012). Conversely, integration tests on large and distributed teams may be complex, and DAD suggests having an independent test team for this (AMBLER; LINES, 2012). The integration tests focus on verifying the potential defects that have fallen through regular unit testing (AMBLER; LINES, 2012). It also aims to execute preproduction testing to ensure quality before a release (AMBLER; LINES, 2012). **Who**: development team, testers, and independent test team. **How**: Brown *et al.* (BROWN; AMBLER; ROYCE, 2013) study presented a set of practical recommendations for achieving improved agility in large-scale software delivery using DAD. Regularly, integration and unit testing occur in parallel, but the authors suggest that integration tests run first. According to the author's suggestion, running integration tests first will demonstrate the architectural challenges and consequently help in resolving the big uncertainties earlier (BROWN; AMBLER; ROYCE, 2013). By putting the economic perspective in the front, resolving unit tests is easier but does not provide economic leverage compared to resolving integration issues first. Since integration tasks are uncertain, if the teams postpone them, they will decrease the probability of success (BROWN; AMBLER; ROYCE, 2013). **Context**: Brown *et al.* (BROWN; AMBLER; ROYCE, 2013) proposed a framework that presents practical recommendations for achieving improvements in agility at large-scale software delivery enterprises. The study is based on three foundational principles that can enable success in achieving agility at an enterprise scale: I) Economic governance; II) Measured Improvement; III) Disciplined agile delivery. Based on these principles, the authors define a framework to prioritize agile practice areas. Finally, focusing on economic advantages, the authors believe that executing integration tests first must reduce uncertainties and provide economic leverage.

4.5.7 T-skilled Individuals (1)

Name: T-skilled individuals. **Goal**: in DAD, T-skilled people are called "generalizing specialists", which are cross-functional people with sufficient skills to get the job done (AMBLER; LINES, 2012). Generalizing specialists have one or more specialties, like programming and testing, and they also have a general knowledge of the overall solution delivery process and domain (AMBLER; LINES, 2012). **Who**: cross-functional team members. **How**: Lal and Clear (LAL; CLEAR, 2018) presented in their study how a large organization with more than ten teams spread across Australia, the USA, and India has evolved their team members to become T-skilled individuals. Initially, the roles were defined as more structured and highly specialized people, which the authors already called generalizing specialists (LAL; CLEAR, 2018). Further, it evolves towards T-skilled individuals who gain in-depth knowledge of the technology and business domain. **Context**: since the study of Lal and Clear described 15 years of transformation of a traditional agile company to a hybrid approach and then to DAD, it was possible to observe the evolution of the roles and the individuals involved during the whole process until they become T skilled individuals.

4.5.8 Product, Program, and Portfolio Planning (1)

Name: product, program, portfolio planning. **Goal**: DAD framework describes several levels of scope plans, which are portfolio plan, program/product plan, release plan, iteration plan, and daily plan (AMBLER; LINES, 2012). The portfolio planning identifies potential new projects and evaluates the dependencies between the ongoing ones, but more details are beyond the scope of the framework (AMBLER; LINES, 2012). It also happens to program/product plans, which look forwards to two or three releases and business goals that it will accomplish (AMBLER; LINES, 2012). Neither of those plans is the focus of DAD (AMBLER; LINES, 2012). **Who**: the organization, leadership. **How**: in Lal and Clear (LAL; CLEAR, 2018) study, the organization chose to combine the product, program, and portfolio planning practices with multi-stakeholder input and a collaborative approach from the business area and the software engineering sector. Independently whether DAD covers it or not, the company could identify long-term goals, discuss reliable and achievable ideas for development, and focus on product management and market needs (AMBLER; LINES, 2012). Finally, the responsibilities shared between the Sales and engineering sectors encouraged a better link between the necessary development skills and collaborative management (LAL; CLEAR, 2018). **Context**: the study of Lal and Clear (LAL; CLEAR, 2018) described how the regular planning activities of a large organization could be used to strict the relationship between the operational teams and the sales area. Due to it, the teams became closer, and the organization drove the solution to a common vision and roadmap.

4.5.9 DAD Training (1)

Name: DAD training. **Goal**: during the transition to a new framework, it is indispensable to provide training for the team members in the process, practices, and roles of the framework. DAD framework suggests the organization conduct training in several disciplines of software engineering. **Who**: the organization. **How**: in Lal and Clear (LAL; CLEAR, 2018) study, the company has hired the proper founder of the DAD framework to active coaching the teams and facilitate the scaled agile transition. Due to it, the team members reported that the transition went smoothly. Finally, they also reported that the training played an important role in leading the teams to success since they needed guidance from the real world (LAL; CLEAR, 2018). **Context**: the study of Lal and Clear (LAL; CLEAR, 2018) covers a huge company capable of affording training and coaching sessions with the proper founder of the DAD framework. However, since this is not affordable for everyone, many companies must rely on alternative approaches to achieve a similar transition level.

4.6 LESS TAILORING PRACTICES

4.6.1 Community of Practice (CoP) (2)

Name: Community of Practice (CoP). **Goal**: CoPs are not originated from LeSS. However, the framework encourages the concept presented by Wenger *et al.* (WENGER; MCDERMOTT; SNYDER, 2002). CoPs are groups of people who share a concern, hobby, or passion regarding a topic or technology and interact to share and improve their knowledge and expertise in the topic (WENGER; MCDERMOTT; SNYDER, 2002). Organizations must not set CoPs but encourage them and form them by the members due to their needs, and the participation must be voluntary (LARMAN; VODDE, 2016a). **Who**: team members. **How**: in the Paasivaaraand Lassenius (PAASIVAARA; LASSENIUS, 2016) case study at Nokia, the CoPs were not seen. The authors justify it due to the previous waterfall mindset culture that the teams dealt with (PAASIVAARA; LASSENIUS, 2016) and the lack of self-organization. However, in Uludag *et al.* (ULUDAG et al., 2019) multiple-case study, the four products evaluated organized the CoPs for collaboration and information exchange involving technical and business subjects. In the study, the principal CoP was an architectural CoP established by architects and stakeholders to discuss architecture-related topics, make decisions, and discuss design guidelines that affect and influence the feature teams (ULUDAG et al., 2019). Further, the teams created additional CoPs for POs, SMs, and even testers (ULUDAG et al., 2019). Finally, the PO CoPs helped them to coordinate and find a common direction on the enterprise level (ULUDAG et al., 2019), similar to a Portfolio level discussion from SAfe (Leffingwell, Dean, 2023). **Context**: both studies, from Uludag textitet al. (ULUDAG et al., 2019) and Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016), describes complex product being developed by several teams spread across the globe. However, the first study saw the opportunity to form CoPs and build them since the team members could self-organize around the hot topics of their day-to-day activities (ULUDAG et al., 2019). In the second study, the previous traditional mindset prevented the teams from needing to discuss the related topics of their activities to improve on that or gain more knowledge (PAASIVAARA; LASSENIUS, 2016).

4.6.2 Requirement Area (2)

Name: Requirement Area (RA). **Goal**: requirement area is a structure required in LeSS huge and organized around customer-centric requirements (LARMAN; VODDE, 2016a). Each requirement area may be divided based on the product boundaries, and it must have an Area Product Owner (APO) with 4-10 feature teams working in the same backlog (LARMAN; VODDE, 2016a; ULUDAG et al., 2019; PAASIVAARA; LASSENIUS, 2016). **Who**: the organization, teams, and APOs. **How**: in the case study of Uludag *et al.* (ULUDAG et al., 2019), the RAs as just described, similar to the LeSS Huge framework (LARMAN; VODDE, 2016a), and it followed the same sprint cadence and the whole areas focused in integrating the entire product through Continuous Integration (CI)/Continuous Delivery (CD) chain. Further, one APO was responsible for each RA and an Area Product Backlog (APB). On the other side, in the case study at Nokia, the 20 teams required requirement areas and APOs responsible for their sprint planning, sprint reviews, and retrospectives (PAASIVAARA; LASSENIUS, 2016). However, the idea of having one APO for each requirement area was not possible in the project since the organization found it difficult to divide the product into different areas (PAASIVAARA; LASSENIUS, 2016). Due to it, several features started to cross several areas, which broke the concept of the LeSS Huge framework (PAASIVAARA; LASSENIUS, 2016).**Context**: the complexity of the project at Nokia and the lack of experience of the organization led them not to apply the requirement area in practice (PAASIVAARA; LASSENIUS, 2016). However, the main consequence of it was related to the fact that several teams were working on the same features, which led to a chaotic atmosphere. Due to the tight schedule, the APOs have given several stories for different feature teams during the same iteration, which led Architect APOs responsible for more than eight teams designing tasks (PAASIVAARA; LASSENIUS, 2016).

4.6.3 Area Product Backlog (2)

Name: area product backlog (APB). **Goal**: APBs are specific product backlog artifacts from requirement areas that the APO maintains (LARMAN; VODDE, 2016a). Each Product Backlog (PB) belongs to one Area Backlog, and vice versa (LARMAN; VODDE, 2016a). **Who**: APOs. **How**: in the study of Uludag *et al.* (ULUDAG et al., 2019), the APO maintains the area product backlog, similar to what the framework says. Further, one of the four products studied had one common backlog and six specific backlogs supported by different area product backlogs (ULUDAG et al., 2019). However, the other three products focused on working on only one product backlog. On the other side, in Nokia's case study (PAASIVAARA; LASSENIUS, 2016), 20 teams were working in a common backlog in an Excel sheet (PAASIVAARA; LASSENIUS, 2016). Further, the solution architect of the APO was responsible for the items in his area and the prioritization suggestion. **Context**: in the study of Uludag *et al.* (ULUDAG et al., 2019), the different nature of the four products evaluated led one of them to establish six specific backlogs due to its complexity. Moreover, in the case of Nokia (PAASIVAARA; LASSENIUS, 2016), with teams spread across four countries, a common backlog was more suitable for their process in the beginning. By the time the project grew, the need for RAs and APB started to appear, and the teams organized through it (PAASIVAARA; LASSENIUS, 2016).

4.6.4 LeSS Huge (1)

Name: LeSS Huge. **Goal**: LeSS provides two different large-scale Scrum frameworks (LARMAN; VODDE, 2016a). Firstly, the LeSS suits one to eight teams with eight people each. Second, the LeSS Huge supports a few thousand people at one product and many teams (LARMAN; VODDE, 2016a). **Who**: the organization. **How**: Uludag *et al.* (ULUDAG et al., 2019) presented a case study of a large company that adopted LeSS in four different products. However, one of the products has 15 feature teams, and they decided to use multiple implementations of LeSS on those teams instead of the LeSS Huge (ULUDAG et al., 2019). The teams' justification was that the subordinate products do not have an overarching product character that would require a LeSS Huge implementation (ULUDAG et al., 2019). From another product perspective, the use of LeSS Huge was more suitable, and the team was having issues with dual leadership by having a PO for the business and another for the IT structure (ULUDAG et al., 2019). To solve this, the organization developed an Area Product Owner (APO) with six product owners responsible for subareas as suggested by LeSS Huge1. **Context**: Uludag *et al.* (ULUDAG et al., 2019) conducted a case study on an automobile manufacturer company to evaluate and report the adoption of LeSS in four products. Even though the study was conducted in the same company, the products were so different that the adoptions generated very different insights regarding the tailoring of practices, roles, and processes.

4.6.5 Single-Specialist Teams (1)

Name: single-specialist teams. **Goal**: in LeSS, the scrum teams are supposed to be cross-competence teams with multi-disciplinary skills (LARMAN; VODDE, 2016a). It also explicitly states that single-specialist teams may not exist since the team should have all the necessary competencies (LARMAN; VODDE, 2016a). However, in the literature, we saw specialist teams aiming to work on very specific parts of the solution and on specific phases of the development lifecycle (PAASIVAARA; LASSENIUS, 2016). **Who**: the organization. **How**: in Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016) study, 20 teams with more than 170 members spread across India, and Europe were trying to scale agile by adopting the LeSS framework. However, single-specialist teams were required despite the regular scrum teams with 6-10 people being multi-disciplinary. First, a CI team was settled in Finland to handle the delivery pipeline, testing environments, and test runs (PAASIVAARA; LASSENIUS, 2016). Further, the company established verifications teams in Germany and Greece to deal with network checks and performance testing (PAASIVAARA; LASSENIUS, 2016). Moreover, the authors reported the presence of a software architecture team with nine specialists responsible for supporting the teams in architecture planning (PAASIVAARA; LASSENIUS, 2016). Finally, a maintenance team composed of two managers and 1-2 members from each team is built. This team aims to take care of trouble reports from external customers and internal platforms (PAASIVAARA; LASSENIUS, 2016). **Context**: despite the recommendations of LeSS Framework (LARMAN; VODDE, 2016a). The complexity of the case study of Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016) required the organization to build multi-disciplinary scrum teams and also work with single-specialist teams. The study was conducted at Nokia, involving people from India, Germany, Finland, and Greece, developing many different products for the telecommunication industry. The complexity of some parts of the solution required single-specialist teams (PAASIVAARA; LASSENIUS, 2016).

4.6.6 Inspect and Adapt (1)

Name: inspect and adapt. **Goal**: inspect and adapt are particularly a practice of SAFe (Leffingwell, Dean, 2023), and according to it, inspect and adapt meetings must focus on evaluating the current status of the product in development and discuss future adaptations for the solution. **Who**: team members, POs. **How**: in the case study presented by Uludag *et al.* (ULUDAG et al., 2019), two of four products that were adopting LeSS conducted regularly inspect and adapt meetings. However, both products choose to adopt the practice with a different purpose, checking where they were in the adoption process, how they could improve this process, and how to tailor their behavior during the adoption (ULUDAG et al., 2019). Those products did not use the practice to check their solution, but two improved the general adoption of LeSS. **Context**: the two out of four products that applied the inspect and adapt practice in Uludag *et al.* (ULUDAG et al., 2019) sutdy, were using LeSS with SAFe (Leffingwell, Dean, 2023) and LeSS with DevOps (EBERT et al., 2016). Due to the combination of LeSS and other frameworks and the lack of previous knowledge in agile methodologies, the teams have seen the inspect and adapt meeting as an opportunity to check whether they were going in the right direction and what they needed to change to keep the adoption going.

4.6.7 Design And Requirement Workshops (1)

Name: design and requirement workshop. **Goal**: design and requirement workshops are helpful to clarify story aspects for the teams (LARMAN; VODDE, 2016a). During the activity, the team members must discuss the inconsistencies the system should support and describe the story through workflow visualizations and abstract descriptions (LARMAN; VODDE, 2016a). **Who**: APO's and team members. **How**: during the case at Nokia from the study of Paassivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016), the design and requirement workshops were arranged by the system and software architects APOs on each user story before the PI Planning, as needed. Moreover, the ideal requirement workshop would discuss the requirement in detail and how the System and Software Architects had decided to implement them at a high-level (PAASIVAARA; LASSENIUS, 2016). On the other side, during the design workshop, the team would plan with the help of architects how to implement the story and deal with its dependencies (PAASIVAARA; LASSENIUS, 2016). However, in reality, the user stories were so big that were requiring those workshops always and took three sprints to be implemented (PAASIVAARA; LASSENIUS, 2016). After fixing this, small stories were gathered into one collaborative design and requirement workshop that could deal with them at once. **Context**: the teams at Nokia had reported that those workshops were helpful since they improved the communication between the APO and them (PAASIVAARA; LASSENIUS, 2016). The teams also said the APO could hear their opinions regarding the design. However, in a rush to release new features, some of them were misplanned in the workshops, which caused issues while implementing, like new requirements or forgetting of planning details (PAASIVAARA; LASSENIUS, 2016).

4.6.8 Retrospective Meeting (1)

Name: retrospective meeting. **Goal**: in LeSS (LARMAN; VODDE, 2016a), the retrospective meetings are supposed to be held as individual meetings on each team, similar to a one-team Scrum retrospective. Further, the team must discuss their issues and large obstacles impeding all the teams (LARMAN; VODDE, 2016a). LeSS also has overall retrospectives with team representatives, scrum master, POs, and managers to discuss general actions and issues (PAASIVAARA; LASSENIUS, 2016). **Who**: team members, POs, SMs, POs, managers. **How**: the Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016) study at Nokia has shown several applications of retrospective meetings. In the beginning, each team had their retrospective meeting and needed to describe three issues that were big enough for they (PAASIVAARA; LASSENIUS, 2016). During the joint retrospective meeting, a discussion would occur based on three issues reported by the teams. However, the problems were too big to be solved in one iteration, and the solution implementation did not follow up (PAASIVAARA; LASSENIUS, 2016). Due to it, the organization tried implementing a different common sprint retrospective with an internal coach to avoid past mistakes. In this new away, the team members must brainstorm the most important impediments, choose the most important one, search for root causes, brainstorm solutions, choose one, and draft implementation of it (PAASIVAARA; LASSENIUS, 2016). Despite the two different approaches, the members' participation was voluntary, and the team members lost interest during the meeting with only a few participants. **Context**: the case study at Nokia involved 170 members spread across four countries and 20 teams (PAASIVAARA; LASSENIUS, 2016). Due to the complexity of solving and fixing the general issues, the teams quickly lost interest in the common meetings, like the retrospective one. The teams also saw the common retrospective meeting as a waste of time since it did not give a big picture of the solution and did not help with coordination issues.

4.6.9 Definition of Done (DoD) (1)

Name: definition of done (DoD). **Goal**: similar to other frameworks, the DoD in LeSS describes the necessary list of criteria that the software needs to meet for each product backlog item (LARMAN; VODDE, 2016a). The DoD must be applied for every product backlog item, and teams must define it in the first Sprint and refine it during the next ones (ULUDAG et al., 2019). **Who**: teams. **How**: in Uludag *et al.* (ULUDAG et al., 2019) case study, each of the four products evaluated establishes a DoD, although not all of them followed the required steps. Only one product finished the DoD specification and followed it during the development (ULUDAG et al., 2019). Two other products defined the first version of their DoD but left it in the corner without proper use by the teams, which was problematic (ULUDAG et al., 2019). Moreover, the case study has shown other definition techniques, such as the Definition of Entry (DoE) and the Definition of Ready (DoR). The DoE describes rough requirements into individual stories, while DoR describes the last step before implementation that represents user stories ready for implementation (ULUDAG et al., 2019). **Context**: Uludag *et al.* (ULUDAG et al., 2019) study present different products with different level of maturity regarding process practices. The rush of day-to-day activities led the teams to ignore the use of DoD during the development, and the organization failed to require it. However, it also led the teams to develop tailored definitions to represent readiness and entries (ULUDAG et al., 2019).

4.6.10 Demo Presentation (1)

Name: demo presentations. **Goal**: demonstration presentations aim to show stakeholders, POs, and APOs the current status of the solution through the execution of some test cases (LARMAN; VODDE, 2016a). **Who**: team members, POs, APOs, stakeholders, and other interested people. **How**: the study of Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016) has described the evolution of a simple demo presentation activity. In the beginning, a common demo meeting of two hours took the place of a common sprint review (PAASIVAARA; LASSENIUS, 2016). Firstly, the teams were gathered in an auditorium, and the representative of each team would share a short slide presentation of the team's achievements. Later, the teams presented real test cases and test results for ten minutes (PAASIVAARA; LASSENIUS, 2016). However, the teams criticized this approach since it would not show the real status of the software and did not encourage discussions or feedback. Further, the organization started to conduct individual demo presentations at each team involving the program manager and PO (PAASIVAARA; LASSENIUS, 2016). **Context**: the case study at Nokia from Paasivaaraand Lassenius has shown how the numbers of teams lead the teams to tailor the demo presentation practice (PAASIVAARA; LASSENIUS, 2016). Since the number of teams evolved, the common meeting became impossible to be held in two hours, and the individual one seemed more efficient.

4.6.11 Scrum of Scrums (SoS) (1)

Name: Scrum of Scrums (SoS). **Goal**: Scrum of Scrums is a meeting usually held by teams at the requirement area level to coordinate their activities (LARMAN; VODDE, 2016a). Commonly, each team sends a representative to discuss and explore cross-team topics, dependencies, and issues on a cadence that can be daily or two to three times a week (LARMAN; VODDE, 2016a). **Who**: team's representatives, scrum masters. **How**: Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016) have presented how the teams at Nokia established a Scrum of Scrums meeting. After their regular daily, one team representative may participate in the daily SoS that would take 5-15 minutes (PAASIVAARA; LASSENIUS, 2016). At the beginning of the project, a common SoS was held in the main site, Finland, although with new teams from India, the main teams tried to include them through teleconference. Since this approach did not go well, the organization split the SoS meeting into two meetings, the first main meeting in Finland and a Global SoS teleconference meeting with teams from all four countries (PAASIVAARA; LASSENIUS, 2016). The project manager was the facilitator, although, during that time, many teams reported nothing in the SoS meetings, and later some representatives started to miss the meetings (PAASIVAARA; LASSENIUS, 2016). **Context**: the case study at Nokia (PAASIVAARA; LASSENIUS, 2016) had 20 teams with 170 members spread across India, Finland, Germany, and Greece (PAASIVAARA; LASSENIUS, 2016). In the beginning, a common SoS meeting seemed enough, although, by the time the teams have grown, some alternatives may take place to handle the dynamic of many people. Further, the company had to deal with the challenge of engaging those people in reporting their problems and issues and keeping their interest in the meeting (PAASIVAARA; LASSENIUS, 2016).

4.6.12 Teams Representatives (1)

Name: teams representatives. **Goal**: many meetings from LeSS are usually held in common with the different teams (LARMAN; VODDE, 2016a). Further, not all members must attend these meetings, and representatives are elected to participate (LARMAN; VODDE, 2016a). **Who**: team's representatives. **How**: in the case study of Paasivaara and Lassenius at Nokia, the (PAASIVAARA; LASSENIUS, 2016) the rotation of teams representatives had two approaches. First, a fixed team representative approach, in which one member of the team or the scrum master is sent to the common meetings (PAASIVAARA; LASSENIUS, 2016). Second, a rotating system was implemented where each team member could participate in the week's common meetings. **Context**: the approach of having common meetings led the 20 teams from Nokia to combine different strategies to have at least one member representing the team at those events (PAASIVAARA; LASSENIUS, 2016). Further, some representatives started to miss the common meetings, which indicates that fixed team representatives can be seen as a burden for some members (PAASIVAARA; LASSENIUS, 2016).

4.6.13 Sprint Planning (1)

Name: sprint planning. **Goal**: LeSS framework defines sprint planning as two-part event (LARMAN; VODDE, 2016a). In the first stage, all teams or representatives organize a common sprint planning. Further, the first stage focuses on selecting ready items presented by the PO, and the teams decide which items they will work on (LARMAN; VODDE, 2016a). In the second phase, each team conducts its sprint planning, in which they create their plan to get the items done during the sprint (LARMAN; VODDE, 2016a). **Who**: PO, SM, and all teams.**How**: Paasivaara and Lassenius's study at Nokia (PAASIVAARA; LASSENIUS, 2016) has shown some adaptions to the sprint planning event. Similar to LeSS (LARMAN; VODDE, 2016a), the company split the event into two parts. First, a common sprint planning for all teams is settled, in which each team sends a representative for a one-hour teleconference meeting (PAASIVAARA; LASSENIUS, 2016). In this meeting, the PO would discuss the market situation and present and assign the user stories for the teams. After it, the sprint planning continues in each team space for detailed planning (PAASIVAARA; LASSENIUS, 2016). Finally, in the evening, each team sent an email with the committed items to the PO and program manager, who updated the backlog (PAASIVAARA; LASSENIUS, 2016). **Context**: the common sprint planning was conducted at Nokia due to a large number of teams involved (PAASIVAARA; LASSENIUS, 2016). However, the team perception has varied. Some saw the meeting as a waste of time, while others enjoyed it since it gave them visibility regarding other teams' work.

4.6.14 Release Planning (1)

Name: release planning. **Goal**: since the LeSS framework is based on Scrum (LARMAN; VODDE, 2016a), the releases must be planned during the product backlog refinement of each iteration. Further, the releases must occur during each iteration.**Who**: the teams, PO, and APOs. **How**: in the study of Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016) at Nokia, the releases suffered a transformation until they arrived at an agile approach. In the first and second years of development, the customers received few versions for test usage (PAASIVAARA; LASSENIUS, 2016). After it, the company established two major software releases per year and six maintenance releases with no new functionality or bug fixing. Meanwhile, the project became more mature and releases once per month for the main customers (PAASIVAARA; LASSENIUS, 2016). **Context**: in this study at Nokia (PAASIVAARA; LASSENIUS, 2016), we must consider the nature of telecommunication projects. Since it's a more legacy industry, the releases used to happen every two or three years. However, the current market perspective expects to access new products and software at a quicker frequency. Due to the project's complexity in the study (PAASIVAARA; LASSENIUS, 2016), the first releases passed through a more traditional approach. However, by the time the project started to mature, the teams and the organization had more courage to establish a shorter release cycle, which the customers received well.

4.6.15 Area Product Owner (APO) (1)

Name: area product owner (APO). **Goal**: according to the LeSS framework (LARMAN; VODDE, 2016a), an Area Product Owner focuses on a customer-centric area and acts as PO concerning the teams of that area. Further, the APO works similarly to the PO but with a more limited perspective since it focuses on a customer-centric area (LARMAN; VODDE, 2016a). APOs can extend to a team of POs or APOs. In those teams, the APO and the PO form a team that makes product-wide prioritization decisions (LARMAN; VODDE, 2016a). **Who**: POs, APOs. **How**: in the Nokia case study presented by Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016), the project had 9-10 APOs responsible for ten different product areas. Further, those APOs were filled by two roles: the solution architect and the system architect (PAASIVAARA; LASSENIUS, 2016). The System architect belongs to the R&D organization sector and is responsible for the technical demands and architectural plans (PAASIVAARA; LASSENIUS, 2016). Meanwhile, the solution architect belongs to the product management sector, which can have a business and technical background (PAASIVAARA; LASSENIUS, 2016). In the same study, the APOs formed a team of APOs, each with those two distinct roles representing an APO. The APO is responsible only for the features of its area and works with the teams to develop them (PAASIVAARA; LASSENIUS, 2016). The APO would have a couple of development teams, and each team would be responsible for developing a feature (PAASIVAARA; LASSENIUS, 2016). However, the issues while establishing requirement areas blocked it 4.6.2. **Context**: the use of Less Huge in the study of Paasivaara and Lassenius at Nokia (PAASIVAARA; LASSENIUS, 2016) involving 20 teams required the presence of APOs as suggested by the framework (LARMAN; VODDE, 2016a). However, the complexity of the project domain required a bit of adaptation, adding two different roles to focus on business and technical demands to represent the original APO role from LeSS (PAASIVAARA; LASSENIUS, 2016).

4.6.16 System and Solution Architects (1)

Name: system and solution architects. **Goal**: originally, LeSS (LARMAN; VODDE, 2016a) does not define the system and solution architects' role, although one article combined both to represent APOs. In LeSS, an APO focus on customer-centric requirement areas as a PO for the teams of those areas (LARMAN; VODDE, 2016a). One study **Who**: APO. **How**: in the Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016) case study at Nokia, the APO role was filled by two different persons, the system architect, and the solution architect. Both were working with features from a requirement area and the teams from it. The solution architects worked separately from the team, using another building since they belong to the product management sector (PAASIVAARA; LASSENIUS, 2016). They were responsible for interacting with customers and market area representatives. Finally, requirements are always passed through solution architects, since one feature may touch several product areas, the organization requires some adaptions on the APO regular model (PAASIVAARA; LASSENIUS, 2016). On the other side, system architects also dealt with features that cross several areas. They were closer to the teams in the building, and their day-to-day activities (PAASIVAARA; LASSENIUS, 2016). The system architect was responsible for updating the backlog based on the teams' progress, arranging requirements and design workshops 4.6.7 (PAASIVAARA; LASSENIUS, 2016). **Context**: the number of requirement areas in the project and the variety of product areas and teams responsible for it lead the organization to tailor the regular APO model suggested by the LeSS framework (PAASIVAARA; LASSENIUS, 2016). Since a feature may cover different areas, just a PO with experience in the business domain would not be enough to handle the technical issues those features may arise. Due to it, the company saw the need to split the APO role into two different roles combining professionals with technical and business skills to evaluate better the cross areas' features (PAASIVAARA; LASSENIUS, 2016).

4.6.17 Domain PO (1)

Name: domain PO. **Goal**: originally, LeSS (LARMAN; VODDE, 2016a) does not specify a domain PO role. However, In LeSS, a PO works similarly to the Scrum role, working as a connector between the customer needs and the teams (LARMAN; VODDE, 2016a). **Who**: PO. **How**: in the four products evaluated in Uludag *et al.* (ULUDAG et al., 2019) study, all of them implemented the domain PO role. The role has project management functions, like synchronizing the feature teams and planning their budget and capacities (ULUDAG et al., 2019). Further, they also have responsibility regarding products at the portfolio level. **Context**: the large-scale environment of the products evaluated at the Uludag *et al.* study (ULUDAG et al., 2019) required a specific professional to handle project management activities regarding budget and capacity tracking. Due to it, a spin from the PO role called Domain PO was born specifically to deal with those subjects preventing the POs from handling one more responsibility (ULUDAG et al., 2019).

4.7 SCRUM TAILORING PRACTICES

4.7.1 Daily Scrum Meeting (25)

Name: daily scrum meeting. **Goal**: in the original Scrum guide, the daily scrum meeting is normally a 15-minute meeting for the team members to inspect progress toward the sprint goal, adapt the Sprint Backlog, and plan for the next day of work (SCHWABER; SUTHERLAND, 2020). It should be held at the same time and place every working day of the Sprint. Also, POs and SMs must participate whether they are actively working on backlog items as team members (SCHWABER; SUTHERLAND, 2020). **Who**: team members, SMs. **How**: due to the distribution of the large projects presented in the sample of this study, most of them implemented the daily meetings through phone and video conferences and sometimes used screen sharing (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; BASS, 2016b; NYRUD; STRAY, 2017; HOSSAIN; BANNERMAN; JEFFERY, 2011; LEE; JUDGE; MCCRICKARD, 2011; MATTHIESEN; BJØRN, 2017; GARBAJOSA; YAGÜE; GONZALEZ, 2014; PAASIVAARA; LASSENIUS, 2010; VALLON et al., 2013; BASS, 2015; LEE; YONG, 2010; WILDT; PRIKLADNICKI, 2010; KUSSMAUL, 2010; HOLE; MOE, 2008; VÄLIMÄKI; KÄÄRIÄINEN, 2008; DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; VALLON. et al., 2013; RALPH; SHPORTUN, 2013; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b; KORKALA; ABRAHAMSSON, 2007). Moreover, the distribution of those teams led to some timezone issues for the companies, which tried different strategies to involve the entire team in the daily meetings. Some authors have shown the presence of daily meetings during the overlap hours of the team (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; LEE; YONG, 2010; DORAIRAJ; NOBLE; MALIK, 2012). In one case, the daily meetings were held in one month in the morning to accommodate European teams better, and in another month at night to better suit the Indian team (DORAIRAJ; NOBLE; MALIK, 2012). In other studies, the international teams were so big that the organization held two dailies on the same day, first at 8 a.m with the occident teams, then at 6 p.m with the orient teams, (LEE; YONG, 2010). Further, some projects with several teams established daily meetings for each team or each site at consecutive times to allow managers and SMs to participate in more than one (BASS, 2014; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; NYRUD; STRAY, 2017;

HOSSAIN; BANNERMAN; JEFFERY, 2011; VALLON et al., 2013; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). Studies that involved onshore and offshore teams applied different strategies involving or not those teams (BASS, 2016b; NYRUD; STRAY, 2017; KUSSMAUL, 2010; NOORDELOOS; MANTELI; VLIET, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). In one case, a 30-minute daily was divided into 15 minutes for the onshore team and the last 15 minutes for the offshore team (NYRUD; STRAY, 2017). From another view, one study showed dailies only with the offshore team since the onshore team focused on generating and maintaining project specifications (KUSSMAUL, 2010). Some studies also reported the conduction of daily meetings through instant messages at chats or email (PAASIVAARA; LASSENIUS, 2010; KUSSMAUL, 2010; HOLE; MOE, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b; KORKALA; ABRAHAMSSON, 2007) due to language problems (HOLE; MOE, 2008). In an exciting strategy, two case studies reported the presence of daily meetings specifically for testers and only to discuss test results (VALLON et al., 2013; VALLON. et al., 2013). Other studies reported different frequencies for the daily meetings, and one case reported a scrum meeting twice a week focused on risk and exploratory testing findings (GUPTA; MANIKREDDY; ARYA, 2017). The presence of two dailies was also common due to timezone differences in combining Indian, American, and European teams (BASS, 2014), or the number of people involved (HOSSAIN; BANNERMAN; JEFFERY, 2011). Also, two studies reported the presence of dailies every two days at least in one of their projects (HOSSAIN; BANNERMAN; JEFFERY, 2011; RALPH; SHPORTUN, 2013). Moreover, some papers reported the benefits of daily meetings on large-scale projects. First, it encouraged communication among the onshore and offshore teams (PAASIVAARA; LASSENIUS, 2010; DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a)and helped them to communicate better and resolve issues faster (NOORDELOOS; MANTELI; VLIET, 2012). Finally, some studies reported the daily meetings as excessive or not required since some teams were too small for it and worked very close every day (VALLON et al., 2014; HOSSAIN; BANNERMAN; JEFFERY, 2011), or teams involved in generating and maintaining project specifications (HOSSAIN; BANNERMAN; JEFFERY, 2011). **Context**: in the 24 studies that were found adaptations to the daily meeting activity, most of them, due to the distributed context, required the use of digital channels to hold the meeting (LEE; JUDGE; MCCRICKARD, 2011; MATTHIESEN; BJØRN, 2017; GARBAJOSA; YAGÜE; GONZALEZ, 2014; BASS, 2015; WILDT; PRIKLADNICKI, 2010; VÄLIMÄKI; KÄÄRIÄINEN, 2008). However, very large-scale projects, such as the study presented by Lee and Yong (LEE; YONG, 2010) at Yahoo, with different big timezones from three continents, required more than one daily to accommodate teams spread around twelve countries. Moreover, studies involving three continents or more also required tailoring through the meetings schedule due to the timezone differences or the number of people involved (BASS, 2014; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; HOSSAIN; BANNERMAN; JEFFERY, 2011; DORAIRAJ; NOBLE; MALIK, 2012). Further, projects involving onshore and offshore teams have shown that depending on the company's maturity level, the daily could involve or not all the teams (BASS, 2016b; NYRUD; STRAY, 2017; HOSSAIN; BANNERMAN; JEFFERY, 2011; KUSSMAUL, 2010; NOORDELOOS; MANTELI; VLIET, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). Projects that involved the offshore teams in the daily have presented fewer issues regarding this activity (NYRUD; STRAY, 2017). Also, distributed teams that were small or working closer justified the execution of dailies through chat messages (KUSSMAUL, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). However, the daily meetings through chat messages were mostly used in studies involving teams with very different cultures and many language barriers to avoid misunderstandings (PAASIVAARA; LASSENIUS, 2010; KUSSMAUL, 2010; HOLE; MOE, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b; KORKALA; ABRAHAMSSON, 2007). It's also essential to notice that small teams, even working distributed, feel that due to the proximity of the members, the daily meeting frequency could be reduced to 2-3 times a week or even be passed to status meeting through chat depending on the team's focus (VALLON et al., 2014; HOSSAIN; BANNERMAN; JEFFERY, 2011).

4.7.2 Scrum of Scrums (SoS) (14)

Name: Scrum of Scrums (SoS). **Goal**: Scrum of Scrums is the purest way of scaling Scrums among multiple teams. However, it is not a practice described in the original framework, but in most of the scaling ones (HENRIK; ANDERS, 2012; Leffingwell, Dean, 2023; LARMAN; VODDE, 2016a). It works as a synchronizing meeting for team representatives to collaborate among themselves. **Who**: team members, Scrum Master, managers, stakeholders, Product Owners, Proxy Product Owners. **How**: most of the studies that reported the use of SoS were using it similar to a daily meeting, but sometimes at a different frequency, through audio and video conferences answering what the team has done since the last meet, what the teams are planning to work on, and their impediments (BASS, 2014; BASS, 2013; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; LASSENIUS, 2010; LEE; YONG, 2010; VÄLIMÄKI; KÄÄRIÄINEN, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; VALLON. et al., 2013). The team representatives must also align their teams' impediments generated for other teams or if they plan to do it (BASS, 2014; BASS, 2013; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). The team's representatives could be in a fixed role or a rotation function. Other studies had reported the conduction of SoS meetings only with Scrum Masters reporting the iteration progress status (BASS, 2014; BASS, 2013; JHA; VILARDELL; NARAYAN, 2016; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b), sometimes related to impediment metrics, weekly defects, overall sprint plan (JHA; VILARDELL; NARAYAN, 2016), and with the POs presence (BASS, 2014). In one study from Gupta *et al.* cite169, the authors presented an experience report on an organization conducting an agile transformation that implemented three SoS meetings. First, a daily SoS meeting with a Scrum Master Part Product Owner managing it, then a weekly SoS meeting conducted by the Chief Scrum Master, and finally a bi-weekly SoS meeting held by the Chief Product Owner (GUPTA; MANIKREDDY; ARYA, 2017). Each meeting had its goals based on the audience, although the common three questions of Scrum were answered (GUPTA; MANIKREDDY; ARYA, 2017). One study has reported a different use of the SoS meeting. First, it involves the leadership team, and the project manager manages it (GUPTA; JAIN; SINGH, 2018). It was also held in an open space focusing on discussions regarding technical topics rather than dealing with impediments (GUPTA; JAIN; SINGH, 2018). However, this approach did not go well during the time since the meeting lost its effectiveness and became a finger-pointing meeting (GUPTA; JAIN; SINGH, 2018). Rolland *et al.* (ROLLAND et al., 2016) conducted a case study of a large-scale agile project that involved 120 participants that used the SoS meetings as an Architecture meeting discussing even alarm build results. Further, in a study involving two different suppliers, only the main supplier had the SoS meetings, while the additional one was not involved (VALLON et al., 2013; VALLON. et al., 2013). **Context**: in the evaluated studies, very different contexts led the teams to apply and tailor the SoS practice. In Gupta *et al.* (GUPTA; MANIKREDDY; ARYA, 2017) study, the software development factory with teams spread across India, Germany, and the USA applied different SoS meetings due to the tailored roles of the company. Further, companies from more traditional fields, such as industry, manufacturing, and oil and energy, were more inclined to conduct one-way SoS meetings involving only SMs reporting regular metrics (BASS, 2014; BASS, 2013; JHA; VILARDELL; NARAYAN, 2016; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). IT Service companies involved with different suppliers suffered similar challenges, although both studies only showed minimal tailoring passing the meetings to a digital channel (VALLON et al., 2013; VALLON. et al., 2013). In studies of sectors using more traditional approaches, Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) has shown a healthcare project that used the SoS meeting as a technical discussion forum can lead the team to deviate from its purpose. Rolland *et al.* (ROLLAND et al., 2016) has shown an SoS meeting focused on Architecture discussions since the teams were looking to improve knowledge transfer and inter-team coordination process. Finally, very large-scale mature companies with distributed teams just tailored the SoS meeting to the digital channels with video and voice conference (LEE; YONG, 2010; VÄLIMÄKI; KÄÄRIÄINEN, 2008).

4.7.3 Retrospective Meeting (10)

Name: retrospective meeting. **Goal**: according to the Scrum guide, the retrospective meeting aims to plan ways to increase quality and effectiveness (SCHWABER; SUTHERLAND, 2020). The team must inspect how the last sprint went regarding individuals, processes, deliverables, tools, and DoD. By identifying things to improve, the team must work on those improvements in the next iteration and evaluate the implemented actions in the next retrospective (SCHWABER; SUTHERLAND, 2020). **Who**: Scrum Team, PO, and SM. **How**: many ways were chosen by the teams and organization to conduct retrospective meetings, including regular retrospective (BASS, 2014; VALLON et al., 2014; HODA; NOBLE, 2017), common retrospectives with all the team (TENDEDEZ; FERRARIO; WHITTLE, 2018; PAASIVAARA; LASSENIUS, 2010; VALLON et al., 2013; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; VALLON. et al., 2013; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b), retrospectives every second (HOSSAIN; BANNERMAN; JEFFERY, 2011; VALLON et al., 2013; VALLON. et al., 2013) or third sprint (LOUS et al., 2018), and even the absence of the practice (HOSSAIN; BANNERMAN; JEFFERY, 2011). In Bass's case study (BASS, 2014), the teams just conducted a regular Scrum Retrospective by the end of each sprint to understand what had been wrong, what was good, and what could be improved. In one of Vallon *et al.* (VALLON et al., 2014) studies with a single Scrum team, the meeting was seen as an invaluable tool. It made the team, in high-stress times, keep improving their process and reduced the frustration level by letting them speak and propose solutions. Helena *et al.* (TENDEDEZ; FERRARIO; WHITTLE, 2018) study on BBC has presented how closed retrospective meetings were helpful for the crews. It promoted flexibility since the team had dedicated time to gather themselves and honestly reflect on their process and culture. Also, the meeting served for the crews to discuss how their work could contribute to the organization's shared goals (TENDEDEZ; FERRARIO; WHITTLE, 2018). In Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011), multiple case studies implemented different approaches to the retrospective practice. One of the cases started the 5-6 sprints holding retrospective meetings, but the practice was discontinued since the Scrum was working well, and the issues were resolved through the other meetings (HOSSAIN; BANNERMAN; JEFFERY, 2011). Also, the lack of feedback from both sites discouraged the continuity of the retrospective meeting. However, in another case of the same study, each team held a common retrospective meeting at the end of the sprint, and the results were posted on the project wiki (HOSSAIN; BANNERMAN; JEFFERY, 2011). Then, those retrospectives started to be every second sprint when the operation began to run smoothly (HOSSAIN; BANNERMAN; JEFFERY, 2011). With a similar result, Lous *et al.* (LOUS et al., 2018) has shown a case in which the retrospective meetings were held every third retrospective week. In Hoda *et al.* (HODA; NOBLE, 2017) study with 31 practitioners, it was perceived that few new teams performed retrospective meetings, but the experienced teams regularly conducted retrospectives to drive continuous improvement. At Paasivaara *et al.* (PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b) multiple case studies, a common retrospective meeting was held after sprint demos involving all the team through teleconference meetings. In another, the onsite team just arranged a unique retrospective meeting, which was perceived as positive. However, later the onsite and offsite scrum masters conducted the retrospectives between them, discussing possible improvement without the team participation (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). Finally, in Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) involving the same case, the common retrospectives meetings with all the teams were held monthly after two sprints. The meeting had six steps, it included an individual evaluation of the last two sprints and the impact of measures taken. Then, new remarks (positive or negative) were presented, and the remarks were clustered into topics and weighed by team members who had three points to vote on different topics. The top three issues were picked up, and the measures were discussed for improvement (VALLON et al., 2013; VALLON. et al., 2013). **Context**: only one case study reported a fully completed retrospective meeting with the script of what to do and how to improve for the next iterations.Interestingly, the studies of Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) reported some issues regarding the relationship between the main supplier and an additional supplier. Still, the retrospective meeting sounds like a practice well implemented with the involvement of the additional supplier representative in an industrial project. Some distributed projects led the retrospective meetings to be conducted after two sprints, and most of it may occur due to the time required to apply and feel the improvements in a large-scale distributed project (HOSSAIN; BANNERMAN; JEFFERY, 2011; LOUS et al., 2018; VALLON et al., 2013; VALLON. et al., 2013). However, independently of the frequency, without the teams' engagement, the practice can be discontinued very quickly or segregated involving only SMs, and POs (HOSSAIN; BANNERMAN; JEFFERY, 2011). The team's maturity is also a criteria to consider in implementing the retrospective meeting. In Hoda *et al.* (HODA; NOBLE, 2017), few new teams conducted retrospective meetings, while more mature ones used it regularly. Also, independent from the context, some distributed projects had reported huge benefits of retrospective meetings that could empower the teams and also made them improve (VALLON et al., 2014; TENDEDEZ; FERRARIO; WHITTLE, 2018).

4.7.4 Status Dashboard (10)

Name: Status Dashboard. **Goal**: Scrum Guide does not present a specific model for progress tracking through a dashboard, like Kanban (SCHWABER; SUTHERLAND, 2020). However, the large-scale distributed projects saw the need to provide the progress and status of the projects in a broader view to all the teams, the management, and even the stakeholders and customers(TENDEDEZ; FERRARIO; WHITTLE, 2018; GUPTA; MANIKREDDY; ARYA, 2017; HODA et al., 2010; GUPTA; JAIN; SINGH, 2018; GODOY et al., 2019; GUPTA; VENKATACHALAPATHY; JEBERLA, 2019; LEE; JUDGE; MCCRICKARD, 2011; VALLON et al., 2013; RAHY; BASS, 2019; VALLON. et al., 2013). **Who**: Scrum Team, POs, SMs. **How**: Many studies using Scrum slipped to Kanban to obtain a dashboard model to provide progress information to stakeholders and teams (TENDEDEZ; FERRARIO; WHITTLE, 2018; GODOY et al., 2019; RAHY; BASS, 2019). Most of them used a digital kanban dashboard which helped the organization to reflect on the teams' overall project status and be up-to-date with the stories' progress without disturbing the workflow (GODOY et al., 2019; RAHY; BASS, 2019). In Helena *et al.* (TENDEDEZ; FERRARIO; WHITTLE, 2018) case study, the organization did not obligate a standard kanban board among the teams. Due to it, teams used physical and digital dashboards simultaneously. Such an approach raised issues since the boards were not synchronized in the same level of richness, and the digital dashboards became a board just for quick progress (TENDEDEZ; FERRARIO; WHITTLE, 2018). On the other side, many studies developed a Scrum Board, similar to some Kanban boards, but with different purposes. Gupta *et al.* (GUPTA; MANIKREDDY; ARYA, 2017) presented a study with two teams spread across Germany and India using a physical dashboard called Wagon Wheel that concentrated the whole progress status of the project on one page. The dashboard was shared during teleconference meetings by the PO and SM (GUPTA; MANIKREDDY; ARYA, 2017). Hoda *et al.* (HODA et al., 2010) multiple-case study has shown that some teams used electronic dashboards, which helped track user stories and tasks. The electronic board enabled collaboration among distributed agile teams since it encouraged to release, iteration planning, and daily stand-ups (HODA et al., 2010). Other studies have used tools in a broader way to embrace another level of information tracking. Similar to an APM solution, in Gupta *et al.* (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019) study, the teams used a tool for monitoring the health and status of the production code. The dashboard is automated and generated from operation scripts. Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) case study, the company used a specific tool called Obeya Wall that concentrated digital and physical dashboards regarding the whole production chain. The tool helped establish communication among the leadership team, product team, distributed stakeholders, and management. Later, the tool gathered performance information, technical debts, pain areas, feedback, customer requests, and quality status (GUPTA; JAIN; SINGH, 2018). In very different applicability, Lee *et al.* (LEE; JUDGE; MCCRICKARD, 2011) study has shown an online discussion board that was used to track the project's issues. It was a central location where the issues could be asynchronously identified, tracked, and addressed. It was percevid as an improvement over emails since the issues could be lost there (LEE; JUDGE; MCCRICKARD, 2011). Finally, in Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) case studies with main and additional suppliers, the additional supplier applied a common virtual dashboard for all three Scrum teams since they were separated from the main supplier. **Context**: it is a consensus that the distributed nature of the studies empowered the organizations to use digital dashboards to provide a big picture of the project progress for interesting parts. However, companies with products that have minimal resistance to issues, like the asset performance management product from Lee *et al.* (LEE; JUDGE; MCCRICKARD, 2011) study, apply a specific board just for issues tracking to map it and solve the issue quickly. In a similar environment, Gupta *et al.* (GUPTA; JAIN; SINGH, 2018; GUPTA; VENKATACHALAPATHY; JEBERLA, 2019) studies involving healthcare solutions opted for the use of potential APM solutions to provide information not only regarding progress status but performance, the health and quality of the production code, technical debts, and customer requests. Such an approach describes the solution's maturity, and the organization's concern with maintaining it (GUPTA; JAIN; SINGH, 2018; GUPTA; VENKATACHA-

LAPATHY; JEBERLA, 2019). The studies that opted for virtual Kanban or Scrum boards just relied on an easier approach to provide visibility for all the teams that compose software factory, and IT service providers company (HODA et al., 2010; GODOY et al., 2019; RAHY; BASS, 2019). The studies that used physical dashboards had the privilege of having collocated teams among the distributed environment of their projects (TENDEDEZ; FERRARIO; WHITTLE, 2018; GUPTA; MANIKREDDY; ARYA, 2017). However, this approach led the teams to share the dashboard through videoconference with remote teams (GUPTA; MANIKREDDY; ARYA, 2017) or to deal with issues in synchronizing physical and digital boards (TENDEDEZ; FERRARIO; WHITTLE, 2018). Further, the suppliers' relationship from *et al.* (VALLON et al., 2013; VALLON. et al., 2013) studies have presented the results of working in an environment with the feeling of "us and them".

4.7.5 Planning meeting (9)

Name: planning meeting. **Goal**: the planning meeting can be considered the starting point of what will be developed in a sprint (SCHWABER; SUTHERLAND, 2020). The Product Owner is responsible for presenting the backlog items to the attendees and discussing them. The most important attendees are the Scrum team, which can invite others to attend looking for advice (SCHWABER; SUTHERLAND, 2020). The event focus on the PO proposing how the product could increase its value. After that, the Scrum team must select items from the product backlog to include in the current sprint. Then, for each item, the Scrum team should plan the necessary effort and work to develop the increment according to the DOD defined in the team (SCHWABER; SUTHERLAND, 2020). **Who**: Scrum team, PO, SM, stakeholders. **How**: several approaches were used to tailor the planning meeting. Most of the studies observed chose to split the meeting into two or three parts by first presenting the backlog items supposed to be implemented in the sprint, then planning in each site or each team, and finally, a final alignment among the teams (HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010; VALLON et al., 2013; HOLE; MOE, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; VALLON. et al., 2013; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). In their multiple-case study, Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011) presented a daily among onshore and offshore teams divided into three parts. First is a teleconference meeting with the PO to review and prioritize the backlog items (HOSSAIN; BANNERMAN; JEFFERY, 2011). Second, due to the timezone, the planning would continue with the offshore team members detailing the tasks. Finally, on the next day, they would present the results to the onshore team, responsible for verifying the plan, estimation, and providing feedback (HOSSAIN; BANNERMAN; JEFFERY, 2011). In another case from the same study, a pre-planning meeting was held involving the PO and SM responsible for prioritizing, assigning, and pre-estimating the items for the teams (HOSSAIN; BANNERMAN; JEFFERY, 2011). In Paasivaara *et al.* (PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b) cases, the teams split the planning into three parts, but with some differences. For maximum effectiveness, the onsite and offsite teams used their three hours overlap in a distributed meeting with the PO for questions and doubts resolution (PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). After that, a local meeting would occur onsite due to the timezone, and the onsite team would focus on the initial estimation and assignment. Then, the next day, a local meeting at the offsite teams would take place to review the onsite plan (PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). From another perspective, in Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) involving a client, the main supplier, and an additional supplier, the planning was divided into two events and considered tasks for the next two sprints. First, the main supplier and two representatives from the additional supplier took the planning on the main site. At this time, they prioritized the items and pre-estimated it (VALLON et al., 2013; VALLON. et al., 2013). Then, the additional supplier representatives would gather the information and present it to their teams, allowing them to assign the tasks themselves, adjust the estimation without many changes, and then present it back to the main supplier (VALLON et al., 2013; VALLON. et al., 2013). In a simpler approach, Hole and Moe (HOLE; MOE, 2008) presented a multiple case study on three GSD projects. In one of them, the local plans took place first to estimate and assign tasks for the remote team. Then, the remote team would receive the results, break them into sub-task, re-estimate and validate it with the local teams (HOLE; MOE, 2008). During that time, the planning began to be seen as time-consuming, and the POs started to create a principal work plan and document the backlog items in there (HOLE; MOE, 2008). Moreover, a case study in a company from a traditional sector was presented by Using scrum in a globally distributed project: a case study Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a) with a planning meeting split into two parts too. First, a distributed meeting with the PO for backlog explanation, then various site-specific arrangements for estimation without needing review by other teams or all the teams, and manager (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). In a very different approach, Kussmaul (KUSSMAUL, 2010) described an experience report involving a customer, an onshore consultancy company, and an offshore team that worked based on a planning team presented in the consultancy company. The planning team controlled the dynamic by developing and signing a formal proposal regarding the major milestones of the projects through a feature list with a price range. When the supplier and the offshore team implemented the features, they would define the price based on the effort and scope developed (KUSSMAUL, 2010). Finally, in Scheerer and Kude's case study (SCHEERER; SCHIMMER; KUDE, 2014), the planning suffered a top-down approach. A central coordination team was responsible for redefining and reprioritizing the individual teams via a new plan. **Context**:in the cases involving many onshore and offshore teams, we could visualize a kind of a pattern regarding the planning meeting despiting the company domain. The companies with many teams spread around several countries were dividing their planning according to the overlap hours of the teams to solve common questions. Then, they allowed the teams to plan on their own according to their timezone, review their estimations and make the necessary adjustments before the sprint begins (HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010; VALLON et al., 2013; HOLE; MOE, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; VALLON. et al., 2013; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). In some of the studies, those types of meetings split into several appointments helped the teams establish a regular discussion forum, cohesion, and identification among them (PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). Sometimes the meeting was even collocated, especially in the first planning 4.7.11, but just for the teams relatively close (PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). In cases where the supplier played an important role in a software product company, the presence of a planning team was helpful for the cost management of deliverables to avoid any bad surprises (KUSSMAUL, 2010). However, in cases where the remote or the offshore team was seen as less skilled related to the onshore team, the remote team didn't have much power to discuss estimation and assignments, which can harm the agile development process (HOLE; MOE, 2008). Finally, more established sectors, such as the one presented by Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a) from an oil and energy company, have shown more maturity in leaving the teams to conduct their specific planning alone after the distributed meeting with the PO.

4.7.6 Multiple Communication Modes (9)

Name: multiple communication modes. **Goal**: since Scrum was designed for collocated teams, the most common type of communication for regular scrum teams is face-to-face communication (SCHWABER; SUTHERLAND, 2020). However, distributed teams in large-scale scenarios require some adaptation, introducing multiple communication modes to handle the distance among the work colleagues. **Who**: team members. **How**: nine studies reported using multiple communication modes to accommodate their different needs of communication better. In general, a wide range of tools and channels are used to support multiple communication modes and substitute face-to-face contact, including phone, web camera, teleconference, video conference, web conference, net meeting, email, shared mailing lists, chats, wiki, ad hoc conversations, and desktop sharing (BASS, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; NYRUD; STRAY, 2017; LEE; JUDGE; MCCRICKARD, 2011; PAASIVAARA; LASSENIUS, 2010; KORKALA; PIKKARAINEN; CONBOY, 2009; HOSSAIN; BABAR; VERNER, 2009; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b; KORKALA; ABRAHAMSSON, 2007). In Lee *et al.* (LEE; JUDGE; MCCRICKARD, 2011) study, the company encouraged the use of asynchronous tools between development and usability teams since their work schedule did not overlap. In the search for effectiveness, the usability engineer becomes available to answer doubts regarding mockups through email and instant chats for the development team (LEE; JUDGE; MCCRICKARD, 2011). Paasivaara *et al.* (PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b) studies had reported using different communication tools based on the need, such as chat messages used for short questions or checking the availability of a colleague for a phone conference. Further, Hossain *et al.* (HOSSAIN; BABAR; VERNER, 2009) presented the project manager as responsible for providing enough communication and collaborative tools for the teams. **Context**: despite the different business domains of those nine studies that used multiple communication modes, all of them have one thing in common, the teams were spread around the globe, and due to it, they required the use of multiple communication modes to handle the teams day to day activities better (BASS, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; NYRUD; STRAY, 2017; LEE; JUDGE; MCCRICKARD, 2011; PAASIVAARA; LASSENIUS, 2010; KORKALA; PIKKARAINEN; CONBOY, 2009; HOSSAIN; BABAR; VERNER, 2009; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b; KORKALA; ABRAHAMSSON, 2007). Only two studies reported the presence of small-scale case projects, although the team members' distribution required the use of asynchronous tools (PAASIVAARA; LASSENIUS, 2010; KORKALA; ABRAHAMSSON, 2007). Further, the large-scale studies involved two to seven teams working around Europe, America, India, and Asia, with very different languages and few overlapping hours. Due to it, the teams required many collaborative tools, such as emails, chat messages, desktop sharing, and others (BASS, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; NYRUD; STRAY, 2017; LEE; JUDGE; MCCRICKARD, 2011; KORKALA; PIKKARAINEN; CONBOY, 2009; HOSSAIN; BABAR; VERNER, 2009; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b).

4.7.7 Product/Project Manager in Scrum (8)

Name: Product/Project Manager in Scrum. **Goal**: product and project managers are not common roles of scrum teams. However, the presence of those professionals was inevitable in some of the reviewed studies. In Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) study, each role had serious responsibilities and goals. First, the project manager was responsible for overall project delivery, interacting with external stakeholders and the Scrum Masters. Second, the product manager was responsible for the product and its business success. **Who**: Product/Project Manager. **How**: most studies combined product or project manager roles to accomplish a better management level in large-scale distributed scrum teams (BASS, 2013; GUPTA; JAIN; SINGH, 2018; HOSSAIN; BABAR; VERNER, 2009; HOLE; MOE, 2008; KORHONEN, 2009; CHO, 2007). Bass (BASS, 2013) presented the presence of a project manager responsible for collecting and prioritizing requirements from different areas of the company to develop and discuss a six-month roadmap. The project manager was also perceived as a client for the development team since he acted as a channel between them and the market (BASS, 2013). From another study perspective, Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) presents different functions for the product and project management professional. First, the project manager would work on defining the overall project plan and quarterly scope, supporting teams' activities as CoPs, removing blockers, and listening to the teams' problems (GUPTA; JAIN; SINGH, 2018). Second, the product manager would focus on feeding business and market requirements to the product, designing business plans with stakeholders and end customers, and interacting with POs without getting involved in day-to-day activities (GUPTA; JAIN; SINGH, 2018). Hossain *et al.* (HOSSAIN; BABAR; VERNER, 2009) reported in the study a project manager with particular responsibilities, such as maintaining policies for GSD teams, ensuring project plans, hiring and forming experienced and effective teams, providing the necessary infrastructure for the team, and following the defined process. Far from what the scrum guide states (SCHWABER;

SUTHERLAND, 2020), Hole and Moe (HOLE; MOE, 2008) reported a project manager responsible for assigning tasks to the team development team. Meanwhile, Cho (CHO, 2007) reported a project manager working on backlog refinement, breaking tasks that were constantly not completed. Moreover, in Korhonen case study (KORHONEN, 2009), three organizations have shown the presence of a project manager as the overall responsibility of the project, including having enough information regarding the project's defects. In Usman *et al.* (USMAN et al., 2018) present a case study conducted in Ericsson, in which the project manager was present in each product. Those project managers had regular traditional responsibilities, such as managing development teams across different sites through planning, scheduling, and coordinating the tasks of product customizations (USMAN et al., 2018). Finally, Martini *et al.* (MARTINI; PARETO; BOSCH, 2013) conducted a multiple-case study that also described the presence of a project manager in Scrum projects. The managers were responsible for the performance of the Scrum teams, either at a project or a product level. **Context**: different contexts have shown the presence of a product or a project manager role, passing from less traditional IT service provider companies (BASS, 2013; HOSSAIN; BABAR; VERNER, 2009; HOLE; MOE, 2008) to the healthcare sector (GUPTA; JAIN; SINGH, 2018), and telecommunication sector (KORHONEN, 2009) until mission-critical software organization (CHO, 2007). In Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) and Bass (BASS, 2013) studies, the product manager was similarly perceived as a customer or a representative of it and also the most interested in the product's success. Moreover, the role of a project manager in the Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) study sounds like an operational manager, worried about delivery, scope, team impediments, and reporting the progress of the business areas. However, in Hossain *et al.* (HOSSAIN; BABAR; VERNER, 2009), the project manager has been perceived as a governance professional, worried about the overall process and some human resource management activities. Further, Hole and Moe (HOLE; MOE, 2008) multiple-case study with teams using Scrum for the first time since the use of traditional approaches justifies the assignment of tasks to the group by the project manager. The mission-critical scenario from Cho (CHO, 2007) study also explains the responsibility of a project manager to break down the tasks for the team, even without achieving success in this activity. In Korhonen case study (KORHONEN, 2009), Usman *et al.* (USMAN et al., 2018) study, and Martini *et al.* (MARTINI; PARETO; BOSCH, 2013) study, the project manager had the most traditional approach seen in the agile studies selected since they were responsible for the whole project activities defect management approach, and also team coordination.

4.7.8 Demo presentation (7)

Name: demo presentation. **Goal**: demo presentation commonly occurs during sprint reviews in Scrum when the team presents the results of their work to key stakeholders (SCHWABER; SUTHERLAND, 2020). The PO and the stakeholders may review what was accomplished and give feedback to the team (SCHWABER; SUTHERLAND, 2020). Despite the distribution environment of the studies in this work, demo presentations were still necessary and, due to it, were tailored. **Who**: team members, PO, SM, project managers. **How**: every study that reported the presence of the demo presentation practice used it to look for feedback from the stakeholders or managers, progress reporting, and validation of the deliverables (BASS, 2014; PAASIVAARA; HEIKKILä; LASSENIUS, 2012; TENDEDEZ; FERRARIO; WHITTLE, 2018; NYRUD; STRAY, 2017; ROLLAND, 2016; HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010). Further, due to the distributed environment, the demos were commonly held through video, and phone conferences using screen sharing (HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010). Bass (BASS, 2014), in his study, report a customer demo without many adaptations at the end of each sprint to refine the product based on the client's feedback. Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) described the evolution of the demo from some practitioners. First, the demo occurred every other week in the team spaces in a successively manner to allow the PPOs, stakeholders from the management, and other teams to participate (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). Then, the demo was specifically held with the system architect, a member of the APO role responsible for engaging with the team (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). In Helena *et al.* (TENDEDEZ; FERRARIO; WHITTLE, 2018) at BBC, the company provided a standard process for demos through a single platform for every team. By doing this, the demos provided visibility and coordination while motivating the teams across the large-scale structure (TENDEDEZ; FERRARIO; WHITTLE, 2018). In another study by Nyrud and Stray (NYRUD; STRAY, 2017), the demo involved different areas, such as technical domain experts, test leaders, and business representatives. Moreover, in Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011), the demos with the customer involved only the management team, the PO, SM, and project management through live meetings to check the sprint completeness, identify problems and provide feedback. Finally, Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2010) study has shown in one of the cases demo involving both onsite and offsite personnel. The demos used screen sharing, which occurred as a face-to-face event during visits. **Context**: studies with teams spread across several continents, including

Asia, Europe, and America, rely on the execution of demos through communication tools, live meetings, and video and phone conferences (HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010). By the time the visits were possible, the demos were adapted to be held in the team spaces (PAASIVAARA; LASSENIUS, 2010). Nyrud and Stray (NYRUD; STRAY, 2017) study was carried out in a company involved in the banking, insurance, and even pension sector, which are strongly regulated. Due to it, the whole areas interested in the product were involved in the demo, from technical experts to business representatives (NYRUD; STRAY, 2017). In regular software development organizations, whether consulting or product companies, the demo presentation and the people involved would vary from managers to customers and stakeholders from other areas interested in the outcome. However, they all have a similar purpose, to provide feedback, evaluate the deliverable, and encourage the teams' collaboration (BASS, 2014; PAASIVAARA; HEIKKILä; LASSENIUS, 2012; TENDEDEZ; FERRARIO; WHITTLE, 2018; ROLLAND, 2016).

4.7.9 Wiki as Communication Tool (7)

Name: Wiki as Communication Tool. **Goal**: wikis are commonly used for teams to store information regarding the project or technical specification of the systems (KORKALA; PIKKARAINEN; CONBOY, 2009). Scrum does not specify the use of wikis since the teams are collocated in the same physical space (SCHWABER; SUTHERLAND, 2020), and the team members can present all the information. However, in distributed and large-scale scenarios, team members may not know each other or make contact due to timezone differences (KORKALA; PIKKARAINEN; CONBOY, 2009; LEE; YONG, 2010; WILDT; PRIKLADNICKI, 2010; KUSSMAUL, 2010; DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b; CHO, 2007).**Who**: Scrum Team **How**: many studies used a wiki as a communication tool and not just a place to concentrate knowledge regarding the project, systems, and solutions. Korkala *et al.* (KORKALA; PIKKARAINEN; CONBOY, 2009) conducted a case study in which the wikis were perceived as the most useful communication tool during the project's implementation phase. The wiki concentrated on technical content, and due to its distributed nature, it helped the teams during software development (KORKALA; PIKKARAINEN; CONBOY, 2009). In Prikladnicki and Wildt (WILDT; PRIKLADNICKI, 2010) study at a multinational company, the wiki helped the teams to control the activities planned for each interaction and the product backlog. It also helped them keep discipline and share the status with teams in different timezones (WILDT; PRIKLAD-

NICKI, 2010). For a similar purpose, one of the companies from (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b) used their backlog as a wiki, allowing everyone to access it and follow the project's progress. In the Cho (CHO, 2007) study, the wiki served the developers as a guideline, gathering all the good and bad practices that developers could perform in the company's software. This approach mitigated common mistakes committed to the project in the past and improved the relationship between the two sites (CHO, 2007). One study reported using a wiki through a mailing list, in which the teams were reporting their status at the end of a working day, describing big changes, current issues, and questions for other teams (KUSSMAUL, 2010). Further, in Dorairaj *et al.* (DORAIRAJ; NOBLE; MALIK, 2012) study, the wiki was used to improve team interaction and foment team presence. The wiki gathered pictures from the team members and personnel moments, which helped them improve their relationship beyond their professional interaction (DORAIRAJ; NOBLE; MALIK, 2012). Finally, Lee and Yong presented an experience report on Yahoo (LEE; YONG, 2010) that used the wiki to manage different backlogs from the teams and products. They also used images to emphasize the human element (LEE; YONG, 2010). **Context**: in general, the wikis were used to support information sharing across the large distributed teams that were working together. The studies that presented the use of wikis as backlogs have shown a certain level of maturity in the companies since they were international companies with extensive experience in developing solutions across teams spread in more than two continents (LEE; YONG, 2010; WILDT; PRIKLADNICKI, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). In a similar context, the study of Dorairaj *et al.* (DORAIRAJ; NOBLE; MALIK, 2012) used the wiki to reduce the distance among members by encouraging them to put photos of them in the wiki since colocating the teams was not an option. Teams that used wikis for technical purposes focused on solving tech issues or guiding the teams to avoid coding design problems in the future. This approach can be beneficial, especially in organizations with many systems that need to stay stable (KORKALA; PIKKARAINEN; CONBOY, 2009; CHO, 2007). Finally, the organization that uses a mailing list as a wiki must know it is easy to lose control of email threads when many teams report on it. However, in Kussmaul (KUSSMAUL, 2010) study, along with the project development, the frequency of emails was reduced, and the teams could control the mailing list.

4.7.10 Proxy Product Owner (PPO) (6)

Name: Proxy Product Owner (PPO). **Goal**: according to the Scrum guide (SCHWABER; SUTHERLAND, 2020), the product owner role is responsible for maximizing the product's value by developing, creating, and ensuring the product backlog items for the Scrum team. He also needs to represent the needs of the stakeholders and end users. However, the proxy product owner is a concept inserted through tailored approaches from large-scale agile distributed projects (BASS, 2014; BASS, 2013; PAASIVAARA; HEIKKILä; LASSENIUS, 2012; HOSSAIN; BANNERMAN; JEFFERY, 2011; BASS, 2015; LEHTINEN et al., 2015). Due to the tailored nature, the role received different functions. A PPO should focus on representing a development team on the client's side and also bring the client's perspective to the team (BASS, 2014). **Who**: team members. **How**: in one of the Bass studies (BASS, 2014), a team's onshore and offshore relationship required the presence of a PPO at the onshore client site to represent the offshore team. He interacts with the client's project team and even the client PO (BASS, 2014). In another study, Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) has shown a different presence of PPO role that works with other PPOs and POs by managing big features with other pairs or a couple of small features alone. Those PPOs also have a technical background, and teams demand architectural guidance from them. Still, they arranged backlog grooming and sprint planning and were responsible for validating teams' demos (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). Moreover, similar to a Scrum PO and a PPO, Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011) presented a proxy customer professional that checked the sprint progress and provided feedback to the team. In a similar perspective, but with a different purpose, Lehtinen *et al.* (LEHTINEN et al., 2015) presented a case study on a large production company with teams spread across three European countries. In this case, the PO did not belong to the Scrum team but were isolated customer representatives who occasionally participated in sprint planning and review meetings (LEHTINEN et al., 2015). As an isolated customer representative, the POs did not participate in the regular Scrum events (LEHTINEN et al., 2015). Finally, Bass studies (BASS, 2013; BASS, 2015) involving the PO functions and teams describe the PPO as the role responsible for staying on the client's site during the beginning of a project to become familiar with any special features of the client's requirement. They are also seen as intermediary roles responsible for mitigating domain complexity. They have extensive experience in the system business domain, acting as an interface to senior executives driving large-scale programs, and disseminating domain knowledge to the teams. **Context**:

the context of Bass studies are very similar (BASS, 2014; BASS, 2013; BASS, 2015). The author evaluated the tailored approaches used to accommodate the Scrum Master (BASS, 2014) and the Product Owner team (BASS, 2013; BASS, 2015) on large-scale distributed enterprises. The regular roles were classified into new categories, but the PPO had the most presence in other studies. Similar to Bass, Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) multiple-case study also focused on understanding the scaling of the PO role, the first case shows how the role was adapted to an Area Product Owner (APO) 4.6.15, and the second case to the PPO model. Further, Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011) multiple-case study looked for the tailoring of regular scrum agile practices and discovered a team using the proxy customer role similar to a PPO. Finally, being a customer representative without proper connection with the Scrum teams, as seen in Lehtinen *et al.* (LEHTINEN et al., 2015) study, could not be the best approach. In the study, the PO expectations failed since they were not seen as part of the Scrum teams, and the developers did not know which PO was responsible for which requirements (LEHTINEN et al., 2015).

4.7.11 First collocated Sprint (6)

Name: first collocated Sprint. **Goal**: in the regular Scrum, a sprint is an interval of time representing a team iteration that should be committed to a goal (SCHWABER; SUTHERLAND, 2020). At the end of a sprint, it is expected for the team to accomplish the goal in the form of a software deliverable. Since Scrum was built for small and collocated teams, every sprint must occur in the physical space. However, in large-scale distributed projects, the sprints may occur remotely among the teams, but some companies choose to make the first sprint with the members collocated (PAASIVAARA; LASSENIUS, 2010; BASS, 2015; KOMMEREN; PARVIAINEN, 2007; DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; DORAIRAJ; NOBLE; ALLAN, 2013). **Who**: team members, SMs, POs, stakeholders. **How**: despite the distributed nature of all studies evaluated in this work. Some demonstrate the importance of providing little face-to-face time for their teams and the importance of it in team building (PAASIVAARA; LASSENIUS, 2010; BASS, 2015; KOMMEREN; PARVIAINEN, 2007; DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; DORAIRAJ; NOBLE; ALLAN, 2013). In Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2010) multiple-case study, one study reported the importance of being collocated for the first and second sprints. The members can learn and develop working habits together during the collocated period, which also helped them later when they needed to work in different sites (PAASIVAARA; LASSENIUS, 2010). In Bass (BASS, 2015) study, one participant reported how offshore members went to the onshore site to work with another team and to have the opportunity to work closely with the product owner and business analysts. In the same line of reasoning, Dorairaj *et al.* (DORAIRAJ; NOBLE; MALIK, 2012) study suggested gathering the entire team with the customer to be collocated for the first few weeks of the project to help them build trust and relationships. After it, it would be natural to emerge the bonding between the entire team and the customer (DORAIRAJ; NOBLE; MALIK, 2012). In Kommeren and Parviainen (KOMMEREN; PARVIAINEN, 2007) experience report on Philips, the organization percevid the importance of gathering the teams physically, improving inter-team coordination performance. Moreover, Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a) study suggested collocating the teams not only in the initial iterations but during critical phases of the project, such as major releases iterations. Finally, despite the importance of gathering customers and teams, Dorairaj *et al.* (DORAIRAJ; NOBLE; ALLAN, 2013) presented the concern of a practitioner of sending newly formed teams to travel to other locations to work collocated for a short time before thoroughly distributing them. **Context**: it's unquestionable the benefits of providing face-to-face interaction among the teams, although it is essential to point out that the studies that conducted collocated iterations could afford it during the project, which is not possible for every organization (PAASIVAARA; LASSENIUS, 2010; BASS, 2015; KOMMEREN; PARVIAINEN, 2007; DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a; DORAIRAJ; NOBLE; ALLAN, 2013). In Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2010) study, the authors also informed that the collocated time must not be a short trip but an extended stay to allow the teams to work together in enough time. Only one study reported the importance of staying together during critical phases, which could be more important depending on the business domain, in this case, oil and energy (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). Further, the multiple case studies that suggest collocated sprints dealt with customer-vendor relationships and onshore and offshore teams, which require face-to-face time to develop trust and a collaborative environment (BASS, 2015; DORAIRAJ; NOBLE; ALLAN, 2013). Finally, in teams specialized in the development of electronic products, such as in the case of In Kommeren and Parviainen (KOMMEREN; PARVIAINEN, 2007), gathering members from Asia and Europe helped them in the inter-team coordination activities of hardware products.

4.7.12 Tools for monitoring progress, quality and knowledge (5)

Name: tools for monitoring progress, quality, and knowledge. **Goal**: Scrum does not specify tools to handle progress monitoring, quality, and test coverage, or even for a knowledge base. However, the Scrum Team should evaluate how the last sprint went concerning individuals, process, DoD, interactions, and even tools (SCHWABER; SUTHERLAND, 2020). **Who**: Scrum team, PO, SM, management. **How**: regarding the distributed nature of the projects in this study, some tools were necessary for coordination activities, test coverage monitoring, and even knowledge sharing. Fitzgerald *et al.* (FITZGERALD et al., 2013b) conducted a case study at a regulated company that needed those tools to accomplish its goals. The organization monitored the codebase every four hours, and if any code changes arose, the Bamboo [2] tool would start a new automated build. On this pipeline, the teams were using Ncover [3] to establish test code coverage beyond 80% since anything below it would block the build (FITZGERALD et al., 2013b). This approach ensured the quality of the code and monitored its progress without allowing quality loss. In Nyrud and Stray (NYRUD; STRAY, 2017) study, the authors evaluated inter-coordination mechanisms. The case used Jira [4] as a project management tool for team coordination. The Jira dashboards supported daily events since the onsite teams could navigate through the tasks in progress from the remote teams (NYRUD; STRAY, 2017). In another use of Jira, Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2010) have shown that small projects preferred using Wikis for coordinating their work. In contrast, large projects picked Jira due to the visibility provided for all teams. Further, Välimäki and Kääriäinen presented a case study (VÄLIMÄKI; KÄÄRIÄINEN, 2008) in which the organization opted for the use of an ALM solution since they did not know the status of the project. Through the tool, the managers and teams started using burn-down charts, bug trends, tasks, and test cases, which helped them better communicate the sprint status across the organization. Finally, Cho presented a case study (CHO, 2007) that used Digital.Ai for project management around both sites. Through the tool, developers could see how each project was divided, their progress, the status of each project, who is working on each one, and when they are supposed to be completed (CHO, 2007). **Context**: different scenarios led the teams to use tools for various purposes. However, the large-scale distributed nature of those studies made almost everyone use team

[2] www.atlassian.com/software/bamboo
[3] www.ncover.com
[4] www.atlassian.com/software/jira

coordination and management tool. In the cases of Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2010) and Nyrud and Stray (NYRUD; STRAY, 2017), the organizations selected Jira to handle the activities of teams spread across Europe, India, and Asia. It would be impossible for companies to handle the dynamics without Jira or a similar tool. However, in a regulated environment like the one presented by Fitzgerald *et al.* (FITZGERALD et al., 2013b), the company working with biological technology would always want to ensure quality and test coverage, and Bamboo with NCover provided it. In a similar perspective, Välimäki and Kääriäinen (VÄLIMÄKI; KÄÄRIÄINEN, 2008) study on the automation industry relied on a more robust tool that could cover each phase of the application lifecycle, such as ALM. Finally, in a mission-critical development environment, such as the one presented by Cho (CHO, 2007), a more traditional tool such as Digital.Ai is more suitable.

4.7.13 Weekly status meeting (5)

Name: weekly status meeting. **Goal**: weekly status meeting is not described as a practice in the Scrum Guide (SCHWABER; SUTHERLAND, 2020). However, large-scale projects needed to resolve issues, cross dependencies, and alignment gaps (JHA; VILARDELL; NARAYAN, 2016; HOSSAIN; BANNERMAN; JEFFERY, 2011; GUPTA; MANIKREDDY; ARYA, 2017; WILDT; PRIKLADNICKI, 2010; HOSSAIN, 2019). **Who**: team members, PO, SM, managers, domain experts. **How**: different weekly meetings were perceived in the studies, but the differences were minimal regarding the meeting subject and the personnel involved. Jha *et al.* (JHA; VILARDELL; NARAYAN, 2016) presented an experience report in which the weekly status meeting was regarding test management to discuss related progress, dependency, and issue resolution. The same study has also presented a global leadership meeting weekly focusing on reviewing overall program progress and key impediments (JHA; VILARDELL; NARAYAN, 2016). In this meeting, the status of the scope, schedule, cost, and quality were reported to the senior management, which used this data to drive their decisions (JHA; VILARDELL; NARAYAN, 2016). In an onshore and offshore relationship, Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011) has shown weekly meetings among the team to stay on track, update the offshore team with any changes, and resolve cross-site issues and dependencies. The weekly meeting also served as a proxy for the SoS meeting 4.7.2. Gupta *et al.* cite169 presented an experience report in which the weekly meeting involved team leads, product managers, and subject matter experts but not the development team. Due to it, the organization suffered communication gaps between the dev teams and domain experts, resulting in rework, schedule slippage, poor code quality due to last-minute changes, and customer complaints (GUPTA; MANIKREDDY; ARYA, 2017). Further, in the Prikladnicki and Wildt (WILDT; PRIKLADNICKI, 2010) experience report, the weekly status meeting served as an integration weekly status meeting. As a 1-hour meeting on Monday to ensure the teams were looking at the correct priority list, to update their progress, and push the next integration task from the backlog. Finally, Hossain *et al.* (HOSSAIN, 2019) presented a typical weekly meeting from a practitioner that took long periods to easier the resolution of challenges like communication gaps, team awareness, and morality issues..**Context**: Jha *et al.* (JHA; VILARDELL; NARAYAN, 2016) study was held at Siemens that used a hybrid development model combining traditional and agile practices. Due to it, the presence of weekly meetings regarding specific disciplines, such as tests, were common, and weekly meetings to discuss overall progress, plan, scope, cost, and quality reinforced some traditional areas from the regular project management discipline (JHA; VILARDELL; NARAYAN, 2016). The onshore and offshore relationship seen in Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011) study required the weekly meeting to accommodate the needs of the sites that required synchronization during the development. Gupta *et al.* (GUPTA; MANIKREDDY; ARYA, 2017) experience report has paid the price of not involving the development team in the weekly status meeting, which led to customer complaints. The biggest reason for not involving the dev team seems to be the past custom of the company of working with traditional manners and not viewing the benefits of including the dev team in the activity (GUPTA; MANIKREDDY; ARYA, 2017). Still, in line with traditional approaches, the Prikladnicki and Wildt (WILDT; PRIKLADNICKI, 2010) experience report at a global multinational company has shown the practice of weekly status meetings focused on integration status, which seems to be a remnant from the traditional activities conducted in the past. Finally, similar to the distributed teams, Hossain *et al.* (HOSSAIN, 2019) used the weekly meeting to solve regular communication gaps and tech issues.

4.7.14 Definition of Done (DoD) (5)

Name: Definition of Done (DoD). **Goal**: the definition of done is a formal description of the state of an increment regarding quality measures that define its born (SCHWABER; SUTHERLAND, 2020). The DoD is useful for establishing a pattern and transparency among the teams' work regarding the completeness of their stories. Whether something does not accomplish the criteria of DoD, it cannot be released or presented in the Sprint Review and must come back to the backlog (SCHWABER; SUTHERLAND, 2020). **Who**: Scrum Team. **How**: Gupta *et al.* cite169 presented an experience report involving teams from India, the USA, and Germany, and the organization established a DoD for all Scrum teams. In the DoD checklist, every story must ensure that it has no static analysis error, no memory leak, and no degradation performance, and must be reviewed by experts (GUPTA; MANIKREDDY; ARYA, 2017). Similar to that checklist, Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b) study has shown a DoD checklist consisting of integration tests completed, version reports made, and user guides updated. From another study, Fitzgerald *et al.* (FITZGERALD et al., 2013b) case in a regulated environment described a DoD concept including regulatory compliance, which must represent the satisfaction of the two customers, the end-users, and the regulatory bodies. Matthiesen and Bjorn conducted a case study (MATTHIESEN; BJØRN, 2017) on a large IT company that suffered issues regarding what "done" means since the government customer did not consider some work of the teams as completed. Due to it, the team realized their interpretation of what was done was equivocal and developed a checklist. The checklist consisted of business analysis, design documentation, unit tests, functional test case, code review, no defects, system test, dependencies check, and acceptance tests approval (MATTHIESEN; BJØRN, 2017). Finally, in Badampudi *et al.* (BADAMPUDI; FRICKER; MORENO, 2013) study on an FDA regulated environment, the DoD perception differs from the developers and product managers due to the quality of the requirements. What the developers considered done was not completed by the product managers. **Context**: two out of four studies that implemented the DoD dealt with projects in highly regulated environments. The adaptions made to the Scrum framework used by the company from Fitzgerald *et al.* (FITZGERALD et al., 2013b) study originated a tailored model called R-Scrum (Regulated Scrum), which included a DoD checklist with regulatory items. Meanwhile, Matthiesen and Bjorn conducted a case study (MATTHIESEN; BJØRN, 2017) in a company that developed software for the Danish government. The type of contract between the company and the government pressed the organization to create a stable DoD checklist to avoid misunderstandings and blocking payments (MATTHIESEN; BJØRN, 2017). Meanwhile, the FDA-regulated environment from Badampudi *et al.* (BADAMPUDI; FRICKER; MORENO, 2013) study did not encourage the teams to build a DoD checklist. Instead, they had issues defining the completeness of a task since the product could not be tested, due to FDA regulations, until it was completed. Further, in cases involving regular IT service companies, the DoD checklist aimed to fix code release issues, establish the teams in a standard pattern for completeness, and avoid problems with end-users and customers (GUPTA; MANIKREDDY; ARYA, 2017; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b).

4.7.15 Component Teams x Generalized teams (5)

Name: component teams x generalized teams. **Goal**: scrum teams are supposed to be cross-functional, which gathers all the necessary skills to create value constantly during sprints and achieve the product's goals (SCHWABER; SUTHERLAND, 2020). They also should be self-managing teams and able to decide who implements what, when, and how (SCHWABER; SUTHERLAND, 2020). Meanwhile, component or generalized teams are tailored approaches used by large-scale distributed projects to handle the complexity of products and solutions developed across the globe (PAASIVAARA; HEIKKILä; LASSENIUS, 2012; MARTINI; BOSCH, 2016; KOCH et al., 2014; HOLE; MOE, 2008; SABLIS; SMITE; MOE, 2021). A component team is usually responsible for a component or part of the full product, having ownership over it and also controlling the development of new features on it (PAASIVAARA; HEIKKILä; LASSENIUS, 2012; KOCH et al., 2014; HOLE; MOE, 2008; SABLIS; SMITE; MOE, 2021). Meanwhile, generalized teams look for the whole product with the same ownership, not for only a part of it, and any member could develop and commit new features to the solution (MARTINI; BOSCH, 2016). **Who**: Scrum Team. **How**: Martini and Bosch (MARTINI; BOSCH, 2016) conducted a large multiple-case study that has solely reported the presence of generalized teams. Two out of five companies from the survey that have shifted to agile approaches have changed their component teams to generalized teams (MARTINI; BOSCH, 2016). It meant that anyone in the project could change any part of the code since a feature needed to be implemented. In Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) study, the authors have seen component teams divided based on the system architecture. Those teams had more technical requirements to implement. Consequently, the POs required more technical background (HOLE; MOE, 2008). Further, Hole and Moe (HOLE; MOE, 2008) presented a multiple-case study in which two of the projects were handled across Norway and India. In India, the company had some remote teams for development. Those teams were separated based on the project's responsibilities. Some members focused specifically on the project GUI (HOLE; MOE, 2008). Sables *et al.* (SABLIS; SMITE; MOE, 2021) conducted a multiple case study in two large-scale projects involving a company from Sweden with teams spread across Asia. The authors percevid that component teams had shown a lower coordination work compared to feature teams, and they considered it due to the lower task interdependence (SABLIS; SMITE; MOE, 2021). Finally, Koch *et al.*

(KOCH et al., 2014) presented two cases in small and large Danish companies. In one of the companies, the teams were grouped based on the areas of Java Development. Beginning with reports development, integration services, ERP maintenance, and monitoring (KOCH et al., 2014). **Context**: Martini and Bosch's (MARTINI; BOSCH, 2016) conducted a large multiple-case study, mostly involving companies developing embedded software solutions on specific hardware. During the agile adoption, one manufacturer company from the telecommunication sector migrated to generalized teams and suffered from a lack of expertise in the teams regarding the solution components (MARTINI; BOSCH, 2016). As expected from a generalized team, by the time the teams become responsible for the whole solution, they also lack proper knowledge of specific components that form the company portfolio (MARTINI; BOSCH, 2016). Meanwhile, in component teams, it's essential to be careful of some characteristics. Component teams may require more technical POs when grouped based on the solution architecture, as shown by Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). However, experienced component teams have shown lower coordination work while having higher expertise in coordinating activities than feature teams since they have a superficial knowledge of the project components (SABLIS; SMITE; MOE, 2021). Further, when developing specific products from a market, having component teams helps to evolve the product in particular directions with specialized staff and a vision of progress from each part (KOCH et al., 2014; HOLE; MOE, 2008).

4.7.16 Product Ownership (4)

Name: product ownership. **Goal**: product ownership is not a concept originally described in Scrum, but from XP (BECK; GAMMA, 2000). In XP, product ownership is usually held by the on-site customer, a client representative available for the team on a full-time basis (BECK; GAMMA, 2000). However, the goals of Scrum are basically around a Scrum team, PO, and SM that must have product ownership to deliver constant aggregated value to the customer by incorporating the values of commitment, focus, openness, respect, and courage (SCHWABER; SUTHERLAND, 2020). Meanwhile, product ownership can be described as a state in which every Scrum team must chase to achieve. It means feeling like one of the owners of the product that understand the solution and its domain while having a shared responsibility among the members for constantly chasing the product's success. **Who**: Scrum Team, management, PO, and SMs. **How**: in Bass study (BASS, 2016b) involving 50 practitioners from 9 companies, the author percevid different configurations of which site the product ownership stayed. Sometimes the product ownership stayed offshore, but it's more common in the onshore site. The author explains that product ownership is closely aligned to the application business domain and domain knowledge, which is more common with the customer onshore (BASS, 2016b). Even when the development management stays offshore, the product ownership usually stays onshore (BASS, 2016b). Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) study considered it difficult to implement shared responsibility through Product Ownership since the Proxy Product Owners 4.7.10 had specific products areas to work without getting involved in the whole product. Finally, Bass studies (BASS, 2013; BASS, 2015) that evaluated the role of PO in large-scale distributed projects have shown a limited perspective of product ownership since the case involved local product owners responsible for specific product areas. **Context**: it´s possible to see a pattern in the studies that had shown product owners responsible for specific areas of the product, or product parts (BASS, 2013; PAASIVAARA; HEIKKILä; LASSENIUS, 2012; BASS, 2015). Since this role works closer to the customer, it´s expected for them to show more product ownership through the development process. However, while they worked on slices of the product, they did not build enough responsibility to present ownership to the whole product (BASS, 2013; PAASIVAARA; HEIKKILä; LASSENIUS, 2012; BASS, 2015). Meanwhile, Bass's study (BASS, 2016b) shows different approaches to having product ownership onshore or offshore depending on customer knowledge, the companies sector, and even the maturity of management.

4.7.17 Requirement Workshops (4)

Name: requirement workshop. **Goal**: design and requirement workshops are original from LeSS framework (LARMAN; VODDE, 2016a) and aim to help and clarify story aspects for the teams (LARMAN; VODDE, 2016a). It was possible to identify some Scrum studies with the same purpose (PAASIVAARA; HEIKKILä; LASSENIUS, 2012; NOORDELOOS; MANTELI; VLIET, 2012; DANEVA et al., 2013). The requirement workshops aim to foment discussion regarding technical dependencies, issues from requirements, and architecture design. **Who**: Scrum team, PO, architect. **How**: Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) conducted a study that has shown the presence of requirement workshops conducted by architects for each new user story from the backlog. The architect must explain at a general level what the customer needs and what the teams must develop, providing a preliminary architecture for the feature (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). Whether needed, the requirement workshop could evolve into a design workshop for more detailed planning with the team responsible for the development (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). In Noordeloos *et al.* (NOORDELOOS; MANTELI; VLIET, 2012) study, after the adoption of Scrum, the offshore team, including developers and testers, participated in requirement workshops to share ideas and solutions with everyone. Daneva *et al.* (DANEVA et al., 2013) multiple-case study has also described the presence of requirement workshops for the user and delivery stories. Such an approach helped the vendor to build trust with the clients through workshops combining face-to-face meetings followed by video and telephone conferencing (DANEVA et al., 2013). Finally, similar to the requirement workshop, Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) experience report has shown that PO and architects conducted idea workshops that gathered the teams for brainstorming and shaping new ideas that could benefit the customer. In such a moment, the organization separated a one-day event for the teams' collaboration and knowledge sharing from each other (GUPTA; JAIN; SINGH, 2018). **Context**: the studies of Noordeloos *et al.* (NOORDELOOS; MANTELI; VLIET, 2012) and Daneva *et al.* (DANEVA et al., 2013) had similar environments involving outsourcing software development, consultancy companies, and client-vendor relationship. The requirement workshop practice helped those scenarios to reduce the misunderstandings from customer requests and to ensure that features were going to be developed with fewer risks regarding the business domain and architecture design (DANEVA et al., 2013; NOORDELOOS; MANTELI; VLIET, 2012). In Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012), the case that implemented requirement and design workshop used it as a communication technique for formal communication with the Scrum teams, which improved the collaboration among members and knowledge sharing. In a similar perception, the Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) study has developed the one-day event for ideas workshop especially to foment team coordination among the members and to decompress the teams from the day-to-day activities.

4.7.18 Developers as Scrum Masters and Product Owners (4)

Name: developers as Scrum Master and Product Owner. **Goal**: according to the scrum guide, developers must be the people from the Scrum Team committed to creating any aspect of a usable increment each sprint (SCHWABER; SUTHERLAND, 2020). More than this, they must plan for the sprint, establish and apply the DoD definition, and stick themselves to the sprint goal by adjusting their plan according to the necessary (SCHWABER; SUTHERLAND, 2020). Any other functions, according to Scrum, related to the product, backlog, prioritization of backlog items, removal of impediments, ensuring the Scrum events, and coaching team members are responsibilities for the PO and SM. However, a tailored approach was seen for those functions due to distributed nature and the large-scale context. **Who**: developers. **How**: in Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013), the authors described how two developers from an additional supplier incorporated the role of unofficial SMs and even POs. The main supplier consultancy company and an additional supplier held three projects. However, only POs and SMs were present at the main supplier site, while the additional supplier had no support for those functions (VALLON et al., 2013; VALLON. et al., 2013). Due to it, the additional supplier suffered with alignments, process implementation, and even stories discussion. Based on it, two developers emerged from the additional supplier with more coordination skills than their colleagues to solve it. They take on Scrum roles and travel to the Main supplier site as SMs and POs to attend meetings and discuss user stories (VALLON et al., 2013; VALLON. et al., 2013). They became the unofficial Scrum master-like roles and improved the project's flow of information by taking care of the process implementation and impediments discussion. In Hole and Moe's multiple case studies, (HOLE; MOE, 2008), one of the GSD projects split the project into modules, each with a Scrum Master. On the remote teams' side, the team members became responsible for the specific modules (HOLE; MOE, 2008). However, a scrum master reported that discussions between sub-teams took too much time due to the distribution barrier. Due to it, the SM appointed one of the remote developers as a local SM, and the distributed SM mostly communicated with this person (HOLE; MOE, 2008). The approach reduced the long-time discussions and also defined a focal point on the team for discussions and process clarification. Hoda *et al.* (HODA et al., 2010) conducted a large case study that presented a important scenario. In cases where the customer lacks involvement, the development teams saw the need to act as proxy product owners 4.7.10 to understand the customer needs better, and the development team did it (HODA et al., 2010). The devs with better communication skills were chosen as the PPO and started to be present with the customer to understand better the project flow (HODA et al., 2010). **Context**: the Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) studies presented a very interesting scenario of suppliers' relationships through the development of large-scale distributed projects. Despite the additional team's skill, they will need support during their implementation process and an understanding of the business needs across the user stories. Whether this support is available or not, the necessity will initially show the issues of lacking a PO and SM in a team. The proactivity of members with more communication skills can solve those gaps (VALLON et al., 2013; VALLON. et al., 2013). However, the team can have performance issues while standing out as PO and SMs and avoid working as developers. A similar context showed the presence of a local SM at Hole and Moe case (HOLE; MOE, 2008). Since an SM was having a lot of discussions with the team members, the developer with the most communication skills must emerge as an SM (HOLE; MOE, 2008). Finally, in typical relationships of consultancy firms with customers, some customers do not play an active role during the project (HODA et al., 2010). Due to it, in Hoda *et al.* (HODA et al., 2010), developers must understand the customer domain and process on their own, or the project can suffer without progress.

4.7.19 Technical Debt Awareness (4)

Name: technical debt awareness. **Goal**: technical debts can be described as issues developed in the solution during the development process due to specific conditions that teams passed. Those conditions can be related to environmental aspects, fast-fix releases, missed predictions of the architecture evolution, production errors, and team lack of experience (CHO, 2007). Despite the origin, technical debts must be addressed and resolved along with the sprint (SCHWABER; SUTHERLAND, 2020) by Scrum. Even without delivering proper value to the customer, it would avoid future problems with scaling, quality, and architecture. **Who**: Scrum Team, and SMs. **How**: Daneva *et al.* (DANEVA et al., 2013), in his multiple-case study, had shown that technical debts were related to the situation when the team considered it safe to start the development without focusing on the architecture design. Based on it, the author refers to technical debts as the amount of architecture redesign work that accumulates over time during development (DANEVA et al., 2013). Consequently, new requirements may require architecture redesign to continue the development, which implicitly brings technical debt awareness to the table (DANEVA et al., 2013). Further, in the Gupta *et al.* (GUPTA; MANIKREDDY; ARYA, 2017) study, technical debt awareness was established through regular knowledge sharing sessions for findings. The teams used it to update and improve their skills and resolve issues while providing relevant training in technical debt solving with the scrum teams (GUPTA; MANIKREDDY; ARYA, 2017). In this case study, the technical debts were considered stories, which may accomplish the criteria from the DoD 4.7.14, pass through code review, validate that the solution did not produce another technical debt, and prevent issues in the

other areas of the solution(GUPTA; MANIKREDDY; ARYA, 2017). Validation of technical debts occurred through the automation tests from the workflows. Also, in this case, the organization built a dedicated multi-functional team focused on resolving technical debt stories, called TD team (GUPTA; MANIKREDDY; ARYA, 2017). The TD team gathered architects, functional experts, testers, and a dedicated PO. They had synchronized sprints with the other teams for three weeks, and after the consumption of the debts backlog, the team was dissolved, and the members regrouped with other Scrum teams (GUPTA; MANIKREDDY; ARYA, 2017). Further, in another Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) study, the technical debts were discussed in a one-day event. In this event, the discussion followed technical topics related to usability improvement, development pain areas like project building effectiveness, static code errors, redundant test cases, and architecture decoupling (GUPTA; JAIN; SINGH, 2018). Finally, Sekitoleko *et al.* (SEKITOLEKO et al., 2014) conducted a case study on Ericsson focusing on the challenges related to technical dependencies. At Ericsson, the organization defined two types of technical debts, the planned technical dependencies, and the unplanned technical dependencies (SEKITOLEKO et al., 2014). The managers, program officers, and the PO (SEKITOLEKO et al., 2014) identified the planned ones during the planning phase. Meanwhile, the unplanned would emerge during the development of improper requirements (SEKITOLEKO et al., 2014).
Context: Daneva *et al.* (DANEVA et al., 2013) presented a multiple case study carried out in a large and mature CMM-5 Asian company widely recognized for its excellence and its engagement in outsourced software development. The authors investigated the requirement engineering process in large-scale contexts (DANEVA et al., 2013). During the process, it was possible to identify that outsourcing projects required a domain owner to transfer knowledge from the client to the teams. Even with those professionals, it is impossible to predict architectural impacts for future features, and due to it, the current features can constantly impact the solution (DANEVA et al., 2013). In Gupta *et al.* (GUPTA; MANIKREDDY; ARYA, 2017) study, at an IT service company, the idea of considering any story as delivery stories helped the teams in the technical debt awareness. By doing this, technical debts had no different treatment. They received an estimation and were mapped to workflows, which helped PO and testers to convince managers to include the debts in the iterations. Also, in test case development and automation (GUPTA; MANIKREDDY; ARYA, 2017). Further, the technical debts were so crucial for the teams that their progress and validation results were presented to the stakeholders. In another study from Gupta *et al.*, (GUPTA; JAIN; SINGH, 2018), in the healthcare sector, the event day for technical debts focused specifically on the team pain areas that consequently

would improve the product quality. Further, the case of Sekitoleko *et al.* (SEKITOLEKO et al., 2014) in a telecommunication company has shown how a more mature company dealt with technical debts that generated technical dependencies across teams. Also, how to plan those dependencies to avoid their impact in the iterations (SEKITOLEKO et al., 2014).

4.7.20 Review meeting (4)

Name: review meeting. **Goal**: during the sprint review meeting, the Scrum team must present the sprint results and their progress to key stakeholders and Product Owner (SCHWABER; SUTHERLAND, 2020). The stakeholder and the PO will inspect the outcome of the Sprint and determine future adaptations (SCHWABER; SUTHERLAND, 2020). The Product Backlog may also be adjusted according to the progress achieved (SCHWABER; SUTHERLAND, 2020). In the distributed large-scale projects studied, the review had similar goals, but due to environmental conditions, the practice needed to be tailored by the teams or the organization (PAASIVAARA; HEIKKILä; LASSENIUS, 2012; HOSSAIN; BANNERMAN; JEFFERY, 2011; VALLON et al., 2013; VALLON. et al., 2013). **Who**: Scrum team, Product Owner, Scrum Master, stakeholders, customers. **How**: in Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) case studies, the suppliers' relationship established a different approach for the sprint review meeting. The meeting is held after each Sprint. However, the event is held at the main supplier site due to the relationship between the main and the additional supplier. The additional supplier accesses it through video conference (VALLON et al., 2013; VALLON. et al., 2013). Additionally, one of the SMs from the additional supplier is present at the main site while the rest of the team observes the meeting (VALLON et al., 2013; VALLON. et al., 2013). During the review, the teams should demonstrate developed features and discuss the current product increments (VALLON et al., 2013; VALLON. et al., 2013). In Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) case study, a common sprint review for all the teams is held, in which a representative of each team would briefly describe what the represented team had accomplished in the previous iteration. Finally, in Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011) multiple-case study with four companies has presented different ways of implementing the review meeting event. A regular sprint review is conducted in the oil and energy sector case company. In this case, at the end of a sprint, the offshore team presented what they developed to the onshore team through video conference tools (HOSSAIN; BANNERMAN; JEFFERY, 2011). In another company from the telecommunication sector, a joint sprint review is held at the end of each Sprint. However, during the demo for the customers, only the management team participated, PO, SM, and project manager, instead of the whole Scrum team (HOSSAIN; BANNERMAN; JEFFERY, 2011). In another case from the same study at an IT service provider company, the Sprint review was tailored to a code review process due to the nature of the product in development (HOSSAIN; BANNERMAN; JEFFERY, 2011). Each sub-team had its code reviewed by other sub-team. At the end of the Sprint, the increment version of the solution was passed to the onshore QA team for review (HOSSAIN; BANNERMAN; JEFFERY, 2011). Finally, in the industry case, instead of a formal review meeting, the developed code based was released to the onshore test engineer for acceptance testing (HOSSAIN; BANNERMAN; JEFFERY, 2011). **Context**: Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) studies involving different suppliers had a review meeting that contributed to the presence of another tailored practice, the Developers as Scrum Masters 4.7.18. Since the main supplier would conduct the review on their site without any representatives from the additional supplier, the developers emerged as SMs to participate and also guarantee the presence of their teams just for observation (VALLON et al., 2013; VALLON. et al., 2013). In Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012), the practitioners described the presence of a review meeting as a regular activity that foments the demonstration of the features accomplished. In Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011) study, the presence of different companies has presented a variety of tailoring approaches for the review meeting practice. Only the most traditional sector, the oil and energy market, has shown the execution of the sprint review event regularly involving both teams by demonstrating the developed features through a video conference tool due to the globally distributed environment. Moreover, in the telecommunication sector, which seems to be less traditional, the review meeting demo process involved only the management team, disconsidering the importance of involving all members from the Scrum team (HOSSAIN; BANNERMAN; JEFFERY, 2011). By the way, the perception of the team and the consequence of not involving them were not described by the authors. Finally, two companies used the review meeting practice to establish a code review process among the teams and conduct a functional test battery involving QA teams or a QA member (HOSSAIN; BANNERMAN; JEFFERY, 2011). However, those companies are an IT service provider, and an industry, which represents different sectors that used the review meeting as an activity of the quality team (HOSSAIN; BANNERMAN; JEFFERY, 2011).

4.7.21 Maintenance Team (3)

Name: maintenance team. **Goal**: Scrum guide does not describe specific functions for a Scrum team. However, the Scrum team is responsible for all product-related activities, including the maintenance of the product (SCHWABER; SUTHERLAND, 2020). Based on a similar purpose, some studies established a maintenance team to handle all the requests regarding the product operation (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). **Who**: Scrum Team. **How**: Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a) present two case studies on a large-scale oil and energy company that had a maintenance team. In this case, the maintenance team did not have a separate PO, but all of the five POs involved in the project could give them tasks regarding their products. However, the maintenance team had SM that worked as a PO by coordinating the maintenance requests from the other teams (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). All customers could add new issues to the maintenance backlog through Jira. However, the organization checked the issue initially, they would pass it to the specific product, and the PO would contact the maintenance team after a quick verification of the issue (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a).The maintenance team was the only team that followed a different sprint cycle. While the regular teams developed through a 4-week sprint cycle, the maintenance team worked through a 2-week sprint cycle since the hotfixes were released every two weeks (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). They also had sprint planning sessions for issue selection, in which they left a buffer of 20% of the capacity for handling fast-track issues from the customers. The most experienced members formed this maintenance team since they needed to know the whole product and constantly impacted customer satisfaction (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). In Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2010) case study, a minimum description of the maintenance team was presented. They had a 2-week sprint cycle synchronized with the 4-week sprint cycle from other teams to be able to release fixes to the customers in a faster approach.**Context**: when a product is in production serving many end users, mainly from critical sectors, a maintenance team seems to be a requirement for product health. Large-scale products usually require constant development of distributed and large teams. However, the production version also needs those teams' attention to better split the demands of the business units. The customer, the maintenance team, stands outs as a suitable option. In both studies from Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a), the maintenance team emerged as a support for the operation in production and to keep customer satisfaction high by solving the issues in short cycles showing care for the product quality.

4.7.22 Technical Area Responsible (TAR) (3)

Name: Technical Area Responsible (TAR). **Goal**: Scrum does not describe a specific area responsible for the technical subjects of the team (SCHWABER; SUTHERLAND, 2020). However, the complexity and the degree of innovation of large-scale projects may require technical references to support the team members during the development. **Who**: tech leader. **How**: Moe *et al.* (MOE et al., 2014) conducted a case study on Ericsson within a project that applied a TAR. The TAR was formed by most of the skilled and senior developers of the projects, who know more about the project and technologies used in the project (MOE et al., 2014). The TARs were seen as essential for the cross-functional teams to work, ensuring the quality and safe evolution of the system. TARs supported teams by answering technical questions regarding their subsystems, they also helped them with design activities and code structure acting as a mentor for less experienced teams (MOE et al., 2014). At Ericsson, the TARs had more responsibilities, like code review, identification of quality issues or improvements for the POs, rejection of design proposals, development of guidelines, and prioritization of trouble reports (MOE et al., 2014). Nyrud and Stray (NYRUD; STRAY, 2017) study has shown the presence of a role with similar responsibilities of a TAR, but the organization called it Tech Liaison. The Tech Liaison is responsible for posses technical insights into the entire product portfolio and serves as a link between the different teams (NYRUD; STRAY, 2017). The role promoted inter-team coordination by facilitating large-scale development. The company created the role to maintain consistency across teams and the technical platform all teams were working on (NYRUD; STRAY, 2017). Finally, Helena *et al.* (TENDEDEZ; FERRARIO; WHITTLE, 2018) has described the presence of specific technical leads for the testers and the developers. Different technical leaders managed each group. **Context**: Moe *et al.* (MOE et al., 2014) conducted a case study on Ericsson in a very large-scale distributed project spread across Sweden, China, and Korea with 17 teams around a variety of subsystems related to the project in development.

The TAR was needed to accommodate many of the technical issues that could arise from those 17 teams and put them on track with the design patterns, and quality level of the company development (MOE et al., 2014). Nyrud and Stray (NYRUD; STRAY, 2017) study occurred at a financial company with different services for the market that was trying to improve inter-team coordination mechanisms. The Tech Liaison role suits the environment needed to ensure consistency among the systems and the teams, which helped the organization (NYRUD; STRAY, 2017). Finally, Helena *et al.* (TENDEDEZ; FERRARIO; WHITTLE, 2018) conducted a case study on BBC that had very defined borders in the software engineering disciplines. Due to this, each tech leader had specific members to take care of according to their skills.

4.7.23 Estimation Contracts (3)

Name: Estimation Contracts. **Goal**: Scrum does not describe any guidance regarding contracts between customers and suppliers. However, in large-scale projects, late changes can generate more costs for the client or more effort for the suppliers, which can harm the project budget. To solve that kind of issue, some companies started to work with buffered fixed-bid contracts (HODA et al., 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a) or iteration contracts (BATRA; VANDERMEER; DUTTA, 2011). **Who**: the organization, management. **How**: Hoda *et al.* (HODA et al., 2010) conducted an extensive case study involving practitioners from 16 organizations and four independent case studies, which dealt with fixed-bid contracts and interaction contracts. Fixed-bid contracts can harm suppliers whether their estimation is not precise or if the customer requests later changes. To avoid such problems, the organization in the study used a buffering technique in which the teams added a 20% buffer to the estimated time taken to develop the project or the feature (HODA et al., 2010). Based on it, the contract is drawn on the estimate considering the buffer for a fixed price and scope (HODA et al., 2010). This approach is used mostly to handle the fixes asked by the customer, and the fast-track issues, as seen in the case study of Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). From another view, some practitioners reported the strategy of selling a few iterations to the customer to begin instead of signing for a large project up front (HODA et al., 2010). By doing this, the organizations were selling an agile trial basis for the customers, which helped them in building confidence with the customer and reducing risk (HODA et al., 2010). By the time the customer may use a few iterations, they are offered to buy more and more iterations of features as needed. Further, another approach used by the practitioners was allowing the customer to swap features since they would not need them anymore and could replace them with new ones with equivalent effort but with more value to them (HODA et al., 2010). Finally, Batra *et al.* (BATRA; VANDERMEER; DUTTA, 2011) presented a study in which the authors developed a complex framework for large-scale agile distributed projects. The authors suggested curbing opportunistic behavior and accounting for cost escalations. A large organization must involve a detailed contract agreement with process-heavy change management to minimize late-change requests from the customer (BATRA; VANDERMEER; DUTTA, 2011). **Context**: The tailoring strategy of fixed-bid contracts from Hoda *et al.* (HODA et al., 2010) study focused on solving one issue from their client-vendor relationship: agile methodologies will not ask you how much time you will need to complete the project, but your customer will. Due to it, the organization must map agile practice into customer practices (HODA et al., 2010). Based on the customer orders, the organization used the agile data to estimate it, like team size, velocity, burndown chart, and members turnover (HODA et al., 2010). From another approach, some practitioners interviewed by Hoda *et al.* (HODA et al., 2010) the study chose to involve their clients in the agile philosophy first by introducing them through small contracts of iterations and then encouraging them to buy more iterations whether the results of the beginning iterations were truly valuable (HODA et al., 2010). Further, allowing the customer to swap features shows them that the supplier can change according to the Agile manifesto (BECK et al., 2001). Still, without losing control of the estimated effort, (HODA et al., 2010). In the Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a) study at an oil and energy industry company, the buffered technique allowed to forecast, at some level, the work. Finally, the suggestions offered by Batra *et al.* (BATRA; VANDERMEER; DUTTA, 2011) consist of a theoretical framework developed based on the literature. Still, real experience is much more advantageous in those cases regarding contracts and costs.

4.7.24 Code freeze (3)

Name: code freeze. **Goal**: code freeze is not a technique specified in the Scrum guide (SCHWABER; SUTHERLAND, 2020), but it was seen as essential to handle several distributed teams working on developing a solution. The practice is also used to ensure the quality of a system during customer tests and to avoid fast changes without proper quality review (BASS, 2014; BASS, 2012; LAUKKANEN et al., 2016). **Who**: Scrum Team, management. **How**: to avoid any new updates ahead of customers' demonstrations, one of the practitioners on Bass study

(BASS, 2014) reported the use of code freeze. At the code freeze time, the teams must use the moment to handle merge issues, while the branches would be blocked, and no team members would be allowed to check in any code. In another Bass study (BASS, 2012), one practitioner reported using code freeze before the start of regression tests. Due to it, during three days, cycles of code freezing, regression testing, and bug fixing were common before adding new code to the solution through the continuous deployment pipeline (BASS, 2012). However, this process was handled by a release department, not the particular teams (BASS, 2012). Finally, Laukkanen *et al.* (LAUKKANEN et al., 2016) conducted a case study on Nokia and described how the code freeze practice was not applied correctly in the organization. When a content package was ready by a team, the teams must freeze the source code and could develop only critical bug fixes afterward, but in practice, the code freeze was not respected (LAUKKANEN et al., 2016). In practice, the customer started his trials when the content packages were ready. However, silent bugs that had passed through the gate requirements were being fixed during customer trials (LAUKKANEN et al., 2016). By doing this, customer requests were also urgently fixed during and after the code freeze time to get things done without waiting for the long planning process. **Context**: In Bass's study, (BASS, 2014), the practitioner that reported the presence of code freeze was involved in developing an airline customer service product related to flight booking. In a product development scenario from a critical sector, such as aviation, code freeze, and branch blocking were necessary to ensure quality for the end-user of systems with minimal failure tolerance. In the other Bass study, (BASS, 2012), the product company that reported the three days code freeze combined with regression tests and bug fixing was maintaining a CRM solution. The solution was big enough to require a release department focused on ensuring the release of the CRM product with quality. Finally, the Laukkanen *et al.* (LAUKKANEN et al., 2016) study at Nokia involved four sites spread across three countries. The code freeze was necessary, but the pressure to release new packages to end users were big enough for the teams to ignore the practice rule.

4.7.25 Community of Practice (CoP) (3)

Name: Community of Practice (CoP). **Goal**: the community of practice (CoP) is very common in other agile frameworks (HENRIK; ANDERS, 2012; LARMAN; VODDE, 2016a). However, it is not originally described on Scrum (SCHWABER; SUTHERLAND, 2020). Most of the organizations that use CoPs in their environment are looking to engage and promote team building in their distributed teams through specific forums, mostly technical and separated by discipline, to discuss issues, news of program languages, frameworks, and architectural design (MOE et al., 2014; GUPTA; MANIKREDDY; ARYA, 2017; GUPTA; JAIN; SINGH, 2018). **Who**: Scrum Team, project manager. **How**: Moe *et al.* (MOE et al., 2014) case study at Ericsson involving Swedish and Chinese teams applied tech forums, similar to CoPs. Those CoPs included test, integration, development, and SMs forums (MOE et al., 2014). In Gupta *et al.* (GUPTA; MANIKREDDY; ARYA, 2017) experience report, the testers from all Scrum teams were dealing with many responsibilities, from test automation to developer support. Beyond it, they were responsible for conducting test CoPs among the members (GUPTA; MANIKREDDY; ARYA, 2017). Finally, in another experience report from Gupta *et al.*, (GUPTA; JAIN; SINGH, 2018), the organization of CoPs relied on the project managers. In their many responsibilities, they should also keep the entire project teams together, creating consistency among them and helping in the establishment and support of CoPS (GUPTA; JAIN; SINGH, 2018). **Context**: Gupta *et al.* (GUPTA; JAIN; SINGH, 2018) experience report from a healthcare company has present a more traditional approach in the project. Due to it, the project manager was the most responsible for the operation, working together with the Scum Master (GUPTA; JAIN; SINGH, 2018). Combining it with the traditional health sector, the managers were responsible for the CoPs, while in most IT service companies, the Scrum Team and its tech leaders were responsible (ULUDAG et al., 2019; PAASIVAARA; LASSENIUS, 2016). In Ericsson's case study from Moe *et al.*, (MOE et al., 2014), the self-organized distributed teams started to conduct CoPs by themselves since they had the autonomy to do that. However, the Chinese teams assumed that they rarely participated in the CoPs, which resulted in fewer interactions between them and the Swedish team and harmed their team-building (MOE et al., 2014). Finally, the other experience report from Gupta *et al.* (GUPTA; MANIKREDDY; ARYA, 2017) at an IT service company experienced in software development in which the roles from the Scrum team, testers, and developers, were responsible for participating in the CoPs, while the managers may only facilitate the CoP.

4.7.26 Scrum training (3)

Name: Scrum training. **Goal**: Scrum suggests that the Scrum Master is responsible for coaching the team members in self-management and cross-functionality (SCHWABER; SUTHERLAND, 2020). More than it, they are also responsible for leading, training, and coaching the organization in its Scrum adoption (SCHWABER; SUTHERLAND, 2020). In large-scale distributed projects, the need in agile coaching is still required for more adoption of Scrum (KOCH et al., 2014; LEE; YONG, 2010; HOBBS; PETIT, 2017). **Who**: Scrum Team, SM, PO, external consultant. **How**: in one of the cases from Koch *et al.* (KOCH et al., 2014), the management team decided to receive training during the Scrum implementation on internal and external delivery processes. The team encouraged five project managers to obtain SM certification by doing this. In Hennel and Dobmeier (HOBBS; PETIT, 2017) single case study, the training, and coaching regarding agile methods were perceived as important factors. Chasing the most positive impact, continuous coaching, and improvement become a rule (HOBBS; PETIT, 2017) in the project. Finally, Lee and Yong presented an experience report on Yahoo (LEE; YONG, 2010) has shown that global product teams and international teams received early coaching and training from the corporate Agile group. However, the international ones were unfamiliar initially (LEE; YONG, 2010). **Context**: Lee and Yong presented an experience report on Yahoo (LEE; YONG, 2010) involving multiple teams spread across three continents and more than twelve countries. The company had enough power to empower the teams through a specific corporate agile group sector from the company. Further, Hennel and Dobmeier (HOBBS; PETIT, 2017) presented a single case study on a telecommunication company evaluating the critical success factors of agile management in large-scale distributed projects. Coaching and training are one of those essential factors of success. (HOBBS; PETIT, 2017). Finally, in Koch *et al.* (KOCH et al., 2014) case studies, the companies started by coaching managers in Scrum. However, the adoption of Scrum failed due to barriers at both companies while translating and externalizing tacit knowledge along the development process with the suppliers. Moreover, the authors pointed out that the suppliers were not specialists in agile software development (KOCH et al., 2014).

4.7.27 Area Product Owner (2)

Name: Area Product Owner (APO). **Goal**: an APO is not described in any Scrum documents or the Scrum Guide (SCHWABER; SUTHERLAND, 2020). However, it is a common practice from the fewer framework (LARMAN; VODDE, 2016a), in which an Area Product Owner focuses on a customer-centric area and acts as PO concerning the teams of that area 4.6.15. Even though it is an original practice from another framework, it was perceived in two studies using Scrum. **Who**: PO. **How**: Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) conducted a study to evaluate how the product owner role has been scaled in large-scale distributed Scrum projects. Through the study, the authors percevid a Scrum case similar to the LeSS approach. The Scrum teams were grouped into customer areas, and each area was headed by an APO (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). Like LeSS (LARMAN; VODDE, 2016a), the APO manages an area-specific backlog, and together with the PO, they formed the PO teams (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). The APO is supposed to work with 2-3 teams developing and managing features of one specific product (PAASIVAARA; HEIKKILä; LASSENIUS, 2012). In another study from Moe *et al.* (MOE et al., 2014), the author focused on understanding the role of knowledge networks at Ericsson. During the process, he describes the presence of an APO as a person responsible for a subsystem. The APOs must work closely with Operative Product Owners through a defined hierarchy, in which the APO is responsible for defining what to implement in a broader view. At the same time, the OPO is an essential part of the teams' social networks (MOE et al., 2014). **Context**: Moe *et al.* (MOE et al., 2014) study was conducted in a very traditional company with an extensive experience in the development of new technologies and with more than ten thousand employees. Hierarchy is necessary for those scenarios, the complexity of the systems in development requires subsystems to be formed and, consequently, a division on the PO role to support the teams (MOE et al., 2014). Further, in Paasivaara *et al.* (PAASIVAARA; HEIKKILä; LASSENIUS, 2012) study, one of the cases chose to work with the APO to scale the PO role into large-scale projects as suggested by Larman and Vodde (LARMAN; VODDE, 2008). Due to the project's needs, the role of APO was divided between two persons: a system architect and a solution architect (PAASIVAARA; HEIKKILä; LASSENIUS, 2012).

4.7.28 Behavior Driven Development (BDD) (2)

Name: Behavior Driven Development (BDD). **Goal**: BDD is a not a original Scrum practice (SCHWABER; SUTHERLAND, 2020). However, it's an XP practice that focuses on developing user stories and test scenarios based on the behavior expected by the software. The acceptance criteria will serve as the base for the BDD, which needs to be understandable by the customer and executable by the testers (VALLON et al., 2013; VALLON. et al., 2013). The BDD helps every role by being a common language and reference point for stakeholders, business analysts, developers, and testers. **Who**: Scrum Team. **How**: Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) studies describe an agile approach of Scrum on a real industry project involving the main supplier and an additional supplier. Both suppliers were developing the solution in the same code base using BDD. Both suppliers aim to have an executable human-readable specification in terms of different scenarios for each story, which could help the testers run functional tests or even develop automation. Also, the stakeholders do understand the behavior (VALLON et al., 2013; VALLON. et al., 2013). However, the use of BDD introduced a lot of overhead to the teams since it was underestimated, resulted in broken case tests, and consequently, bad code quality (VALLON et al., 2013; VALLON. et al., 2013). Moreover, testers constantly struggled to finish the automation of BDD scenarios within the sprints, which resulted in issues during delivery (VALLON et al., 2013; VALLON. et al., 2013). **Context**: Vallon *et al.* (VALLON et al., 2013; VALLON. et al., 2013) studies described in rich detail the dynamics of suppliers working in the development of a project for the customer. With good purpose, the teams tried to establish the use of BDD in the development and planning of stories, although it sounds like the teams lacked experience in the applicability of the practice. Most of the issues were related to the underestimation of the scenarios, the lousy quality of the code produced, and the overhead of the testers that did not have enough throughput in automating the tests cases that had the stories developed (VALLON et al., 2013; VALLON. et al., 2013). Finally, it sounds like the team must have a certain level of maturity before adopting a practice such as BDD, which requires enough skill from all roles of the Scrum Team.

4.7.29 Design Pipeline (2)

Name: design pipeline. **Goal**: design pipeline is a specific practice seen in studies with large-scale distributed projects using Scrum. However, it's not a standard practice from Scrum (SCHWABER; SUTHERLAND, 2020) or even any other agile framework. The design pipeline aims to establish a design sprint ahead of the development sprint (HODA et al., 2010; LEE; JUDGE; MCCRICKARD, 2011). The practice's main goal is to reduce developers' overhead by giving them interfaces already built just for their implementation. **Who**: Scrum Team, designers. **How**: in Lee *et al.* (LEE; JUDGE; MCCRICKARD, 2011) case study,a usability engineer work ahead of the development team to develop the design and delivery it for developers to start implementing those designs in the following iterations. This approach helped the team to optimize their velocity and predictability while developing a system that would meet high-level design goals. In similar applicability, Hoda *et al.* (HODA et al., 2010) study involving 40 practitioners from 16 large-scale organizations also reported the presence of a design pipeline. One team adapted the agile practices to fit the context of a front-end design-intensive project.

Due to it, the design activities were running ahead of the development by one sprint (HODA et al., 2010). The main focus of the design pipeline was to support the design tasks in driving the backend functionalities. For each iteration, the designers must have their duties ready by the beginning of the next iteration for the development team (HODA et al., 2010). Further, the design pipeline must handle the zeroth iteration, including only design activities. At the same time, the development team must proceed with their tasks following the front-end designs by one iteration (HODA et al., 2010). **Context**: Lee *et al.* (LEE; JUDGE; MCCRICKARD, 2011) conducted a study at Meridium company with teams across India and USA to evaluate how the custom eXtreme Scenario-based design (XSBD) approach, developed by Virginia Tech for usability centered projects, could be used in a distributed environment using Scrum. The strategy aims to ensure that the interfaces built by the teams agree with the project's high-level goals and the prototypes of the dedicated usability engineer. Due to it, the usability engineer works one iteration ahead of the development teams. While the development team implements a new interface, the usability engineer develops the designs for the next iteration. Meanwhile, in Hoda *et al.* (HODA et al., 2010) study, a designer saw the design pipeline as an adaption for ensuring that developers did not waste substantial effort on technical matters before getting the front-end design. Especially in a project whose context was skewed towards being front-end design-intensive. Finally, looking at both studies that implemented the design pipeline practice, it is possible to conclude that projects with a heavy need for front-end interface development may require a design team focused on developing interfaces ahead of the development team. All of it reduces the developers' overhead and avoids letting them develop interfaces far from the business value expected by the customers and end users.

4.7.30 Futurospective (1)

Name: futurospective meeting. **Goal**: futurospective is not even near being an original practice from Scrum (SCHWABER; SUTHERLAND, 2020). However, it was a workshop event conducted by agile coaches, and a few managers created a vision of where the organization would be in a couple of years (PAASIVAARA et al., 2014). It is a kind of roadmap and vision presentation regarding the organization and its product. **Who**: agile coaches, managers. **How**: the focus of the futurospective meeting is to answer the following question: “What made this product such a huge success?”. According to Paasivaara *et al.*, (PAASIVAARA et al., 2014), based on the question, the coaches and managers started to write a "showcase" (PAASIVAARA et al., 2014). The "showcase" concentrated on a vision of how the organization would look in two years and how the work would be done for the whole organization to create a highly successful product. Based on the "showcase", the organization's values were created. **Context**: Paasivaara *et al.* (PAASIVAARA et al., 2014) conducted a case study at Ericsson to describe how a new organization acquired by Ericsson used "Value Workshop" the different sites and teams that transitioning from plan-driven to lean and agile on a large-scale distributed context. The teams were involved in developing a new product, consisting of three teams spread across two countries in Europe and one in Asia. For better suitability with the Ericsson environment, the teams worked on common values that were originated by their interpretation and behavioral implications as a team from the workshops, and futurospective meetings (PAASIVAARA et al., 2014).

4.7.31 Story Owners (1)

Name: story owner. **Goal**: in Scrum, the PO role has responsibilities regarding the development and communication of the product goal, building and management of product backlog items, order of product backlog items, and ensuring the backlog is transparent, visible, and understood (SCHWABER; SUTHERLAND, 2020). The PO also needs to be the face and voice of the product since the role represents the need of the customer, stakeholders, and end-users (SCHWABER; SUTHERLAND, 2020). However, story owner is a tailored practice to substitute the regular PO role from Scrum (HODA et al., 2010). Instead of being responsible for the whole product, they are responsible for particular stories of less than a week long (HODA et al., 2010). **Who**: story owners. **How**: the concept of story owner was seen in the cases from Hoda *et al.* (HODA et al., 2010). The story owner is responsible for particular stories, so each story from the product backlog has an owner. The idea of working with it had a specific purpose. Even when a single customer representative may be required for a regular project, it's impossible to expect continuous availability from them (HODA et al., 2010). Due to it, the story owners could play this role for a specific time and particular stories, avoiding issues regarding information necessity from the customer (HODA et al., 2010). Such a role becomes more present when customer involvement is not enough for the project, and the supplier or team members cannot change this nature (HODA et al., 2010). The approach also allowed the teams to plan new stories based on the current story being developed together with the story owner. It also allowed the customer representatives to create a sense of ownership among the stories they were responsible for (HODA et al., 2010). **Context**: Hoda *et al.* (HODA et al., 2010) conducted an extensive case study in 16 organizations across India and New Zealand, corroborated with four other case studies. In those cases, the story owner practice proved successful for the practitioners when the teams needed more customer collaboration (HODA et al., 2010). The practice was suitable, especially for projects involving clients and suppliers, where the suppliers require proper customer attention and participation for the project to succeed.

4.7.32 Limited blast radius technique (1)

Name: limited blast radius technique.**Goal**: this practice originates itself in some studies that used the Spotify framework (SALAMEH; BASS, 2019). However, with the same goal of releasing a new version of the solution to a small portion of end-users (BASS, 2012) to avoid high risks of incidents in the whole base of customers. **Who**: team members. **How**: with a similar purpose to the studies that used Spotify Framework (HENRIK; ANDERS, 2012), one of the companies from Bass study (BASS, 2012) reported the use of the limited blast radius technique during the releases of new software increment to selected market to avoid incidents across the whole users (BASS, 2012). Based on it, software was continuously released to a specified number of end-users. However, when an incident occurred, the squad could roll back the changes and stabilize the environment (BASS, 2012) without harming the entire customer community. **Context**: in the case study presented by Bass with seven international companies (BASS, 2012), one company that had business on the internet chose to apply the limited blast radius technique due to the difficulty of getting feedback from their customers, which was facilitated due to release in selected markets on an experimental basis.

4.8 TARGET FRAMEWORK - TAILORING LARGE-SCALE AGILE PRACTICES FRAMEWORK

This section will present the developed TARGET framework based on the tailored practices reported by the large-scale distributed project studies. For this framework, the whole 95 tailored practices identified from the studies using Spotify, SAFe, DAD, LeSS, and Scrum (HENRIK; ANDERS, 2012; Leffingwell, Dean, 2023; AMBLER; LINES, 2012; LARMAN; VODDE, 2016a; SCHWABER; SUTHERLAND, 2020) were organized based on the market sector of the companies studied. In total, 17 different business domains are identified in the selected studies.

Each identified market sector is grouped based on the scale dimension of the teams' and the agile scaling framework used. The main idea of the built framework is to serve as a guideline for practitioners and researchers to understand better how a variety of companies with different business domains have tailored agile practices using different agile and scaling agile frameworks already known in the market.

The company domains and their scale were extracted and grouped while evaluating the studies and developing the tailored practices. A single study can appear in different domains or scaling terminologies since the sample contains multiple-case studies involving various organizations and studies conducted in different years in the same organization.

Finally, the business domains that compose the frameworks are telecommunication, oil and energy, automotive industry, process & industry automation, healthcare, optical industry, financial, science and research, broadcasting, BI and Big Data, Logistic, Mission-critical software, enterprise CRM, general industry, Internet, software service provider, and IT service provider. The following subsections will present each business domain in the framework structure, describing the organizations' scale, the agile scaling framework used, the studies from this domain, and the mapped tailored practices identified. The market sectors within a small number of tailored practices were combined in a specific subsection.

To better understand each sector, the practices presented on it, and the studies which originated them, it was developed a studies map table 4.8 in which each study was classified with an acronym from S1 to S74.

Table 6 – Studies Map.

Name	Study
S1	(ULUDAG et al., 2019)
S2	(EBERT; PAASIVAARA, 2017)
S3	(BASS, 2014)
S4	(PAASIVAARA, 2017)
S5	(BASS, 2013)
S6	(RAZZAK et al., 2018)
S7	(JHA; VILARDELL; NARAYAN, 2016)
S8	(BASS, 2012)
S9	(PAASIVAARA; HEIKKILä; LASSENIUS, 2012)
S10	(PAASIVAARA; LASSENIUS, 2016)
S11	(PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008)
S12	(VALLON et al., 2014)
S13	(SALAMEH; BASS, 2019)
S14	(FITZGERALD et al., 2013b)
S15	(MARTINI; BOSCH, 2016)
S16	(BASS, 2016b)
S17	(MARTINI; PARETO; BOSCH, 2013)
S18	(TENDEDEZ; FERRARIO; WHITTLE, 2018)
S19	(NYRUD; STRAY, 2017)
S20	(ROLLAND, 2016)
S21	(MOE et al., 2014)
S22	(SALAMEH; BASS, 2020)
S23	(HOSSAIN; BANNERMAN; JEFFERY, 2011)
S24	(PANDYA; MANI; PATTANAYAK, 2020)
S25	(LOUS et al., 2018)
S26	(LAL; CLEAR, 2018)
S27	(GUPTA; MANIKREDDY; ARYA, 2017)
S28	(BROWN; AMBLER; ROYCE, 2013)
S29	(HODA et al., 2010)
S30	(HODA; NOBLE, 2017)
S31	(GUPTA; JAIN; SINGH, 2018)
S32	(GODOY et al., 2019)
S33	(GUPTA; VENKATACHALAPATHY; JEBERLA, 2019)
S34	(LEE; JUDGE; MCCRICKARD, 2011)
S35	(MATTHIESEN; BJØRN, 2017)
S36	(LAUKKANEN et al., 2016)
S37	(GARBAJOSA; YAGÜE; GONZALEZ, 2014)
S38	(PAASIVAARA; LASSENIUS, 2010)
S39	(VALLON et al., 2013)
S40	(RAHY; BASS, 2019)
S41	(BASS, 2015)
S42	(KOMMEREN; PARVIAINEN, 2007)
S43	(KORKALA; PIKKARAINEN; CONBOY, 2009)
S44	(KOCH et al., 2014)
S45	(BADAMPUDI; FRICKER; MORENO, 2013)
S46	(SEKITOLEKO et al., 2014)
S47	(LEE; YONG, 2010)
S48	(WILDT; PRIKLADNICKI, 2010)
S49	(HOSSAIN, 2019)
S50	(LAUTERT; NETO; KOZIEVITCH, 2019)
S51	(KUSSMAUL, 2010)
S52	(KORHONEN, 2009)
S53	(HOSSAIN; BABAR; VERNER, 2009)
S54	(HOLE; MOE, 2008)
S55	(VÄLIMÄKI; KÄÄRIÄINEN, 2008)
S56	(DORAIRAJ; NOBLE; MALIK, 2012)
S57	(LEHTINEN et al., 2015)
S58	(PAASIVAARA et al., 2014)
S59	(PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a)
S60	(SABLIS; SMITE; MOE, 2021)
S61	(BEECHAM et al., 2021)
S62	(HOBBS; PETIT, 2017)
S63	(USMAN et al., 2018)
S64	(ROLLAND et al., 2016)
S65	(SCHEERER; SCHIMMER; KUDE, 2014)
S66	(VALLON. et al., 2013)
S67	(DANEVA et al., 2013)
S68	(DORAIRAJ; NOBLE; ALLAN, 2013)
S69	(RALPH; SHPORTUN, 2013)
S70	(NOORDELOOS; MANTELI; VLIET, 2012)
S71	(BATRA; VANDERMEER; DUTTA, 2011)
S72	(PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b)
S73	(CHO, 2007)
S74	(KORKALA; ABRAHAMSSON, 2007)

Source: The author (2023)

4.8.1 IT Service Providers

IT service providers are the most common sector in our study. All the studies conducted on consultancy firms, software factories, and IT outsourcing companies were grouped in this category. 27 studies representing more than a third of the total selected papers for the SLR are part of this sector, also involving three different frameworks, Scrum (SCHWABER; SUTHERLAND, 2020), SAFe (Leffingwell, Dean, 2023), and DAD (AMBLER; LINES, 2012), and with the three scale dimensions, small, large and very large-scale (See table 4.8.1). Most IT service provider companies concentrate on large-scale projects using Scrum. Still, two studies that used SAFe and DAD significantly represented the practices mapped on both frameworks. See figure 4.8.1 for a more visible view of the IT service provider sector framework. The unique study from a small-scale dimension was a case study from Vallon *et al.* (VALLON et al., 2014) that described action research from a single Scrum team through 27 sprints when transitioning to a distributed environment and required the tailoring and scaling of agile practices.

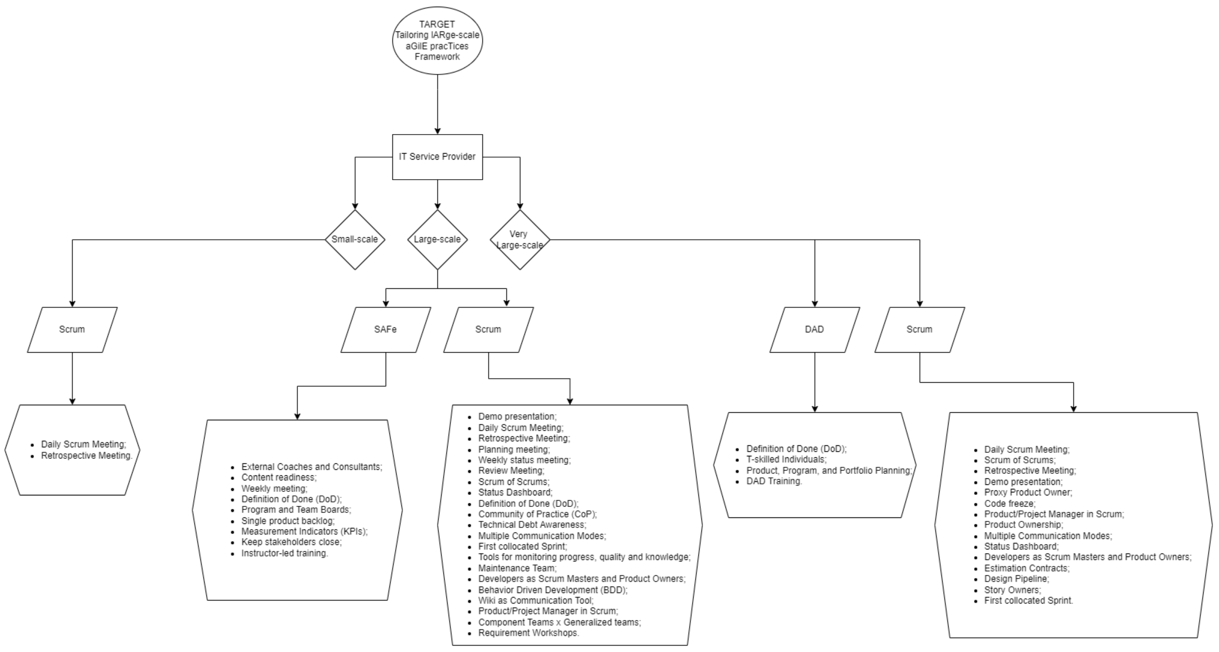
A total of 25 studies from IT service providers used tailored Scrum practices. Those studies used 26 out of 32 practices from Scrum, and only six tailored practices were not present in this sector. The IT service providers' study sample is responsible for almost the whole tailored mapped practices from Scrum. Moreover, Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) experience report described the transition from Scrum to SAFe in large-scale development teams in India, which implemented 9 out of 24 tailored practices from SAFe (Leffingwell, Dean, 2023). Of those nine practices, seven were only present in the Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) study. Finally, Lal and Clear conducted a long case study over more than 15 years regarding the transition from Rational Unified Process (RUP), through a Hybrid-Agile method, to a Scaled Agile approach using Disciplined Agile delivery (AMBLER; LINES, 2012) involving ten teams in a very large-scale context (LAL; CLEAR, 2018). The study has four out of nine practices from DAD, representing almost half of the tailored practices mapped in the framework.

Table 7 – Tailored practices from the It Service Providers sector.

Market sector	Scale	Frameworks	Studies	Practices
IT Service Provider	Small-scale	Scrum	S12	▪ Daily Scrum Meeting (S12) 4.7.1; ▪ Retrospective Meeting (S12) 4.7.3.
	Large-scale	SAFe	S24	▪ External Coaches and Consultants (S24) 4.4.2; ▪ Content readiness (S24) 4.4.3; ▪ Weekly meeting (S24)4.4.7; ▪ Definition of Done (DoD) (S24) 4.4.8; ▪ Program and Team Boards (S24) 4.4.9; ▪ Single product backlog (S24) 4.4.13; ▪ Measurement Indicators (KPIs) (S24) 4.4.14; ▪ Keep stakeholders close (S24) 4.4.15; ▪ Instructor-led training (S24) 4.4.16.
		Scrum	S20, S23, S27, S30, S32, S35, S38, S39, S40, S43, S45, S49, S53, S54, S56, S66, S70, S72	▪ Demo presentation (S20, S38) 4.7.8; ▪ Daily Scrum Meeting (S23, S27, S35, S38, S39, S54, S56, S66, S70, S72) 4.7.1; ▪ Retrospective Meeting (S23, S30, S38, S39, S66, S72) 4.7.3; ▪ Planning meeting (S23, S30, S38, S39, S54, S66, S72) 4.7.5; ▪ Weekly status meeting (S23, S27, S49) 4.7.13; ▪ Review Meeting (S23, S39, S66) 4.7.20; ▪ Scrum of Scrums (S27, S38, S39, S66, S72) 4.7.2; ▪ Status Dashboard (S27, S32, S39, S40, S66) 4.7.4; ▪ Definition of Done (DoD) (S27, S35, S45, S72) 4.7.14; ▪ Community of Practice (CoP) (S27) 4.7.25; ▪ Technical Debt Awareness (S27) 4.7.19; ▪ Multiple Communication Modes (S38, , S53, S72) 4.7.6; ▪ First collocated Sprint (S38, S56) 4.7.11; ▪ Tools for monitoring progress, quality and knowledge (S38) 4.7.12; ▪ Maintenance Team (S38) 4.7.21; ▪ Developers as Scrum Masters and Product Owners (S39, S54, S66) 4.7.18; ▪ Behavior Driven Development (BDD) (S39, S66) 4.7.28; ▪ Wiki as Communication Tool (S43, S56, S72) 4.7.9; ▪ Product/Project Manager in Scrum (S53, S54) 4.7.7; ▪ Component Teams x Generalized teams (S54) 4.7.15; ▪ Requirement Workshops (S70) 4.7.17.
	Very large-scale	DAD	S26	▪ Definition of Done (DoD) (S26) 4.5.3; ▪ T-skilled Individuals (S26) 4.5.7; ▪ Product, Program, and Portfolio Planning (S26) 4.5.8; ▪ DAD Training (S26) 4.5.9.
		Scrum	S3, S5, S8, S16, S29, S41	▪ Daily Scrum Meeting (S3, S16, S41) 4.7.1; ▪ Scrum of Scrums (S3, S5) 4.7.2; ▪ Retrospective Meeting (S3) 4.7.3; ▪ Demo presentation (S3) 4.7.8; ▪ Proxy Product Owner (S3, S5, S41) 4.7.10; ▪ Code freeze (S3, S8) 4.7.24; ▪ Product/Project Manager in Scrum (S5) 4.7.7; ▪ Product Ownership (S5, S16, S41) 4.7.16; ▪ Multiple Communication Modes (S8) 4.7.6; ▪ Status Dashboard (S29) 4.7.4; ▪ Developers as Scrum Masters and Product Owners (S29) 4.7.18; ▪ Estimation Contracts (S29) 4.7.23; ▪ Design Pipeline (S29) 4.7.29; ▪ Story Owners (S29) 4.7.31; ▪ First collocated Sprint (S41) 4.7.11.

Source: The author (2023)

Figure 10 – IT Service Provider visual Framework guide.



Source: The author (2023)

4.8.2 Telecommunication

A total of 13 studies were held in telecommunication organizations representing the second sector with more studies involving three different frameworks, Scrum (SCHWABER; SUTHERLAND, 2020), SAFe (Leffingwell, Dean, 2023), and LeSS (LARMAN; VODDE, 2016a) (See table 4.8.2 and figure 4.8.2). The sector had no studies involving small-scale teams or projects; since telecommunication organizations seem to be large companies, the selected studies involved large-scale and very large-scale distributed projects.

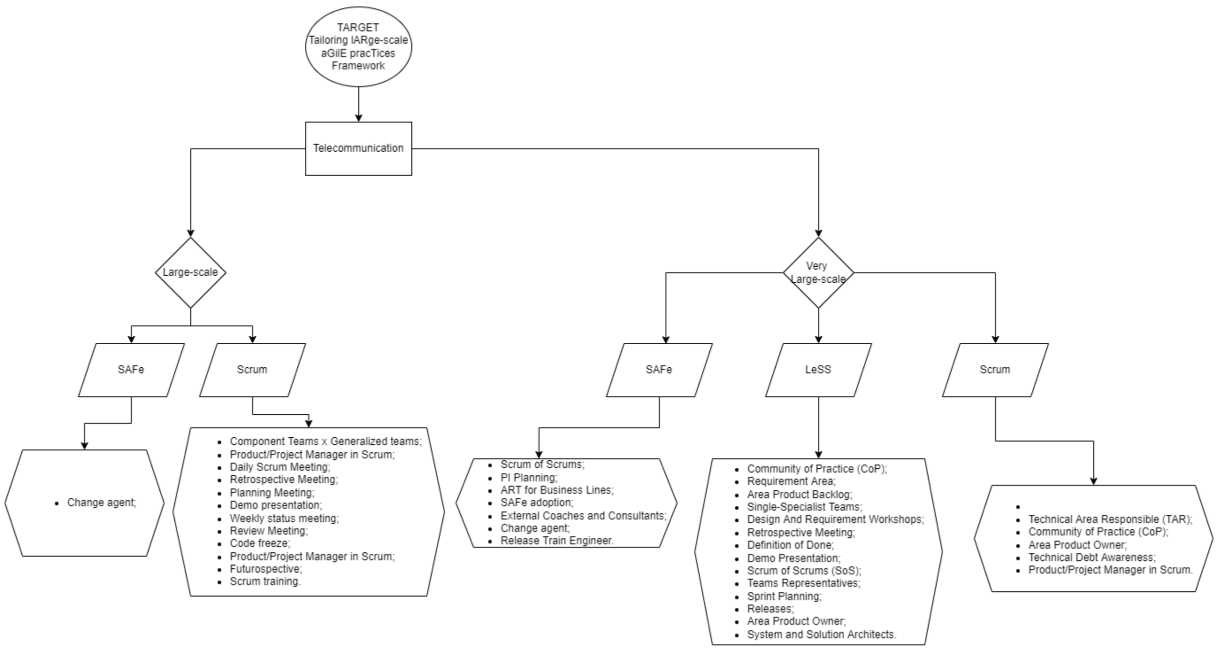
The telecommunication sector concentrates 38 of the whole 95 practices, which represents more than a third of all practices. Most telecommunication studies used Scrum in large-scale and very large-scale distributed environments, with a total of 10 studies using 15 out of 32 Scrum practices, almost half of them. Moreover, the SAFe studies from the telecommunication sector represent eight practices from a total of 24. Finally, the Paasivaara and Lassenius (PAASIVAARA; LASSENIUS, 2016) case study at Nokia that used LeSS framework (LARMAN; VODDE, 2016a) represents the case study with more tailored practices regarding the LeSS framework (LARMAN; VODDE, 2016a), with a total of 14 practices out of 17 practices mapped during the study.

Table 8 – Tailored practices from the Telecommunication sector.

Market sector	Scale	Frameworks	Studies	Practices
Telecommunication	Small-scale	N/A	N/A	N/A
	Large-scale	SaFe	S2	Change agent (S2) 4.4.23;
		Scrum	S15, S17, S23, S36, S52, S58, S62	Component Teams x Generalized teams (S15) 4.7.15; Product/Project Manager in Scrum (S17) 4.7.7; Daily Scrum Meeting (S23) 4.7.1; Retrospective Meeting (S23) 4.7.3; Planning Meeting (S23) 4.7.5; Demo presentation (S23) 4.7.8; Weekly status meeting (S23) 4.7.13; Review Meeting (S23) 4.7.20; Code freeze (S36) 4.7.24; Product/Project Manager in Scrum (S52) 4.7.7; Futurospective (S58) 4.7.30; Scrum training (S62) 4.7.26.
	Very large-scale	SAFe	S4	Scrum of Scrums (S4) 4.4.10; PI Planning (S4) 4.4.1; ART for Business Lines (S4) 4.4.22; External Coaches and Consultants (S4) 4.4.2; Change agent (S4) 4.4.23; Release Train Engineer (S4) 4.4.24.
		LeSS	S10	Community of Practice (CoP) (S10) 4.6.1; Requirement Area (S10) 4.6.2; Area Product Backlog (S10) 4.6.15; Single-Specialist Teams (S10) 4.6.5; Design And Requirement Workshops (S10) 4.6.7; Retrospective Meeting (S10) 4.6.8; Definition of Done (S10) 4.6.9; Demo Presentation (S10) 4.6.10; Scrum of Scrums (SoS) (S10) 4.6.11; Teams Representatives (S10) 4.6.12; Sprint Planning (S10) 4.6.13; Release Planning (S10) 4.6.14; Area Product Owner (S10) 4.6.15; System and Solution Architects (S10) 4.6.16.
		Scrum	S21, S46, S63	Technical Area Responsible (TAR) (S21) 4.7.22; Community of Practice (CoP) (S21) 4.7.25; Area Product Owner (S21) 4.7.27; Technical Debt Awareness (S46) 4.7.19; Product/Project Manager in Scrum (S63) 4.7.7.

Source: The author (2023)

Figure 11 – Telecommunication Framework visual guide.



Source: The author (2023)

4.8.3 General Industry

Many selected studies from the SLR were conducted in industrial environments. Since a study did not describe the type of industry, it was classified in the general industry sector of the framework. A total of 12 studies represent the general industry sector, all of them using and tailoring Scrum practices in the three different dimensions of agile scaling, one with a small-scale project (WILDT; PRIKLADNICKI, 2010), seven with large-scale projects (JHA; VILARDELL; NARAYAN, 2016; MARTINI; BOSCH, 2016; MARTINI; PARETO; BOSCH, 2013; HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010; WILDT; PRIKLADNICKI, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b), and five with very large-scale projects (BASS, 2014; BASS, 2013; BASS, 2012; BASS, 2015; KOMMEREN; PARVIAINEN, 2007) (See table 4.8.3 and see figure 4.8.3). The small-scale project was considered in our results due to the distribution of the team between the USA and Brazil, which led them to tailor agile practices due to their distributed nature and the size of the company involved (WILDT; PRIKLADNICKI, 2010).

The industrial sector studies had tailored half of the Scrum practices mapped in the SLR. 16 out of 32 practices from Scrum are present in those studies. The Daily Scrum Meeting 4.7.1 in the most tailored practice in industries (HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010; WILDT; PRIKLADNICKI, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b), followed by the Planning 4.7.5 and Retrospective Meetings 4.7.3 (HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b).

Table 9 – Tailored practices from the General Industry sector.

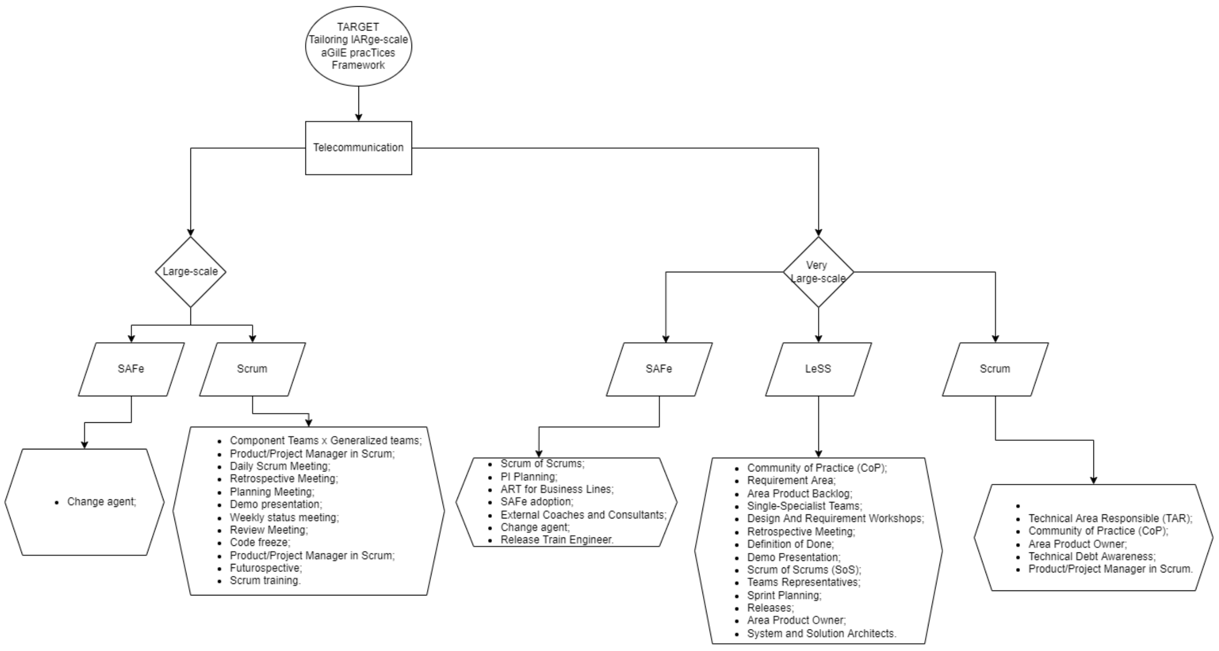
Market sector	Scale	Frameworks	Studies	Practices
General Industry	Small-scale	Scrum	S48	Daily Scrum Meeting (S48) 4.7.1; Weekly status meeting (S48) 4.7.13.

	Large-scale	Scrum	S7, S15, S17, S23, S38, S48, S72	Scrum of Scrums (S7, S38) 4.7.2;
				Weekly status meeting (S7, S23) 4.7.13;
				Component Teams x Generalized teams (S15) 4.7.15;
				Product/Project Manager in Scrum (S17) 4.7.7;
				Daily Scrum Meeting (S23, S38, S48, S72) 4.7.1;
				Retrospective Meeting (S23, S38, S72) 4.7.3;
				Planning Meeting (S23, S38, S72) 4.7.5;
				Review Meeting (S23) 4.7.20;
				Multiple Communication Modes (S38) 4.7.6;
				Demo presentation (S38) 4.7.8;
				First collocated Sprint (S38) 4.7.11;
				Tools for monitoring progress, quality and knowledge (S38) 4.7.12;
				Maintenance Team (S38) 4.7.21;
				Definition of Done (DoD) (S38) 4.7.14;
				Wiki as Communication Tool (S48) 4.7.9.

	Very large-scale	Scrum	S3, S5, S8, S41, S42	Daily Scrum Meeting (S3) 4.7.1;
				Code freeze (S8) 4.7.24;
				Multiple Communication Modes (S8) 4.7.6;
				Daily Scrum Meeting (S41) 4.7.1;
				Scrum of Scrums (S5) 4.7.2;
				First collocated Sprint (S42) 4.7.11.

Source: The author (2023)

Figure 12 – General Industry visual Framework visual guide.



Source: The author (2023)

4.8.4 Software Service Provider

Software service providers configure companies that develop one or more solutions for the market, have their income prevenient from those solutions, and are responsible for maintaining and distributing the software products for the clients. This sector, together with the general industry, is the second sector with more studies in our framework, with a total of 12 studies on it using two different frameworks, Scrum (SCHWABER; SUTHERLAND, 2020) and DAD (AMBLER; LINES, 2012) (See table 4.8.4 and see figure 4.8.4).

Most of the studies from software service providers used Scrum in small, large, and very large-scale projects. Further, only the study from Brown *et al.* (BROWN; AMBLER; ROYCE, 2013) using DAD suggested the use of the Integration and Unit Testing 4.5.6 practice in their framework. Moreover, less than half of the Scrum practices were used in this sector, precisely 14 out of 32 practices from Scrum. Finally, the small-scale project presented in this sector is part of a more extensive study involving large companies in Denmark (KOCH et al., 2014). Further, despite the project's small scale, the members worked in a distributed manner since they were in different countries.

Table 10 – Tailored practices from the Software Service Provider sector.

Market sector	Scale	Frameworks	Studies	Practices
Software Service Provider	Small-scale	Scrum	S44	Component Teams x Generalized teams (S44) 4.7.15; Scrum training (S44) 4.7.26.

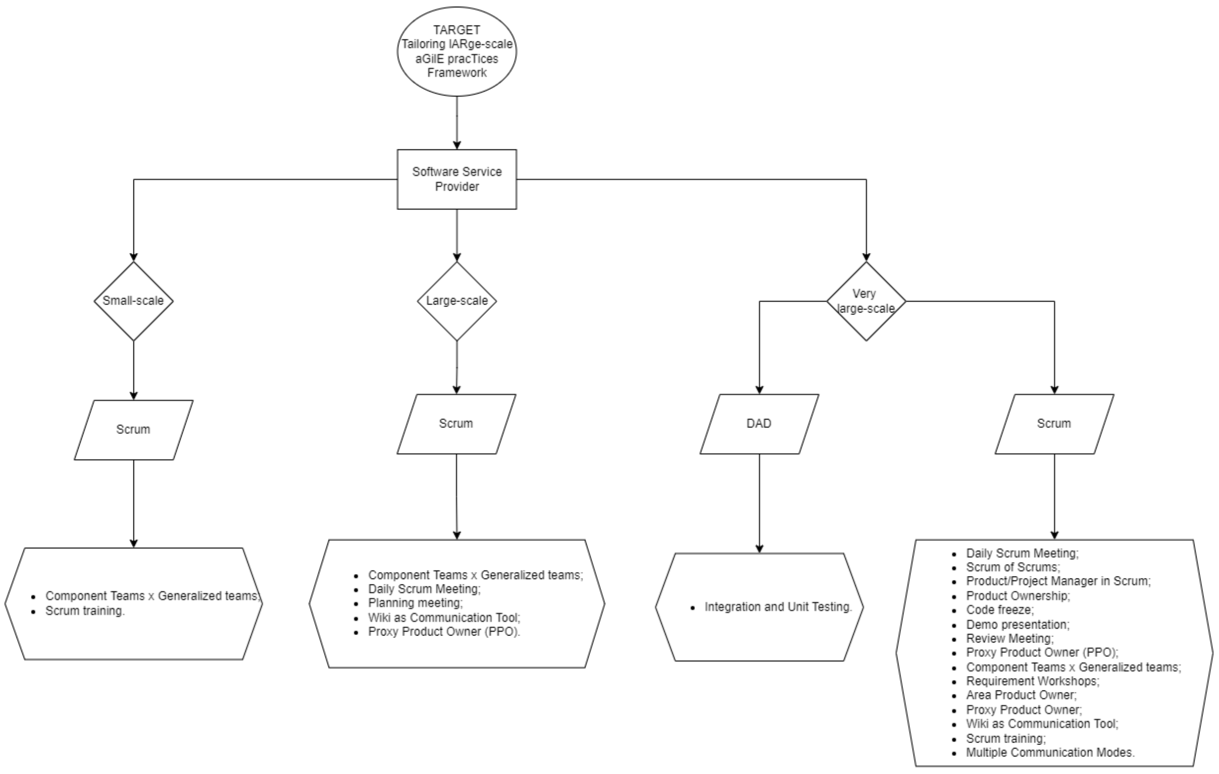
	Large-scale	Scrum	S15, S51, S57	Component Teams x Generalized teams (S15) 4.7.15;
				Daily Scrum Meeting (S51) 4.7.1;
				Planning meeting (S51);
				Wiki as Communication Tool (S51) 4.7.9;
				Proxy Product Owner (PPO) (S57) 4.7.10.

	Very large-scale	DAD	S28	Integration and Unit Testing (S28) 4.5.6.

		Scrum	S3, S5, S8, S9, S41, S47, S74	Daily Scrum Meeting (S3, S41, S47, S74) 4.7.1;
				Scrum of Scrums (S3, S47) 4.7.2;
				Product/Project Manager in Scrum (S5) 4.7.7;
				Product Ownership (S5, S9, S41) 4.7.16;
				Code freeze (S8) 4.7.24;
				Demo presentation (S9) 4.7.8;
				Review Meeting (S9) 4.7.20;
				Proxy Product Owner (PPO) (S9) 4.7.10;
				Component Teams x Generalized teams (S9) 4.7.15;
				Requirement Workshops (S9) 4.7.17;
				Area Product Owner (S9) 4.7.27;
				Proxy Product Owner (S41) 4.7.10;
				Wiki as Communication Tool (S47) 4.7.9;
				Scrum training (S47) 4.7.26;
				Multiple Communication Modes (S8, S74) 4.7.6.

Source: The author (2023)

Figure 13 – Software Service Provider Framework visual guide.


TARGET
Tailoring lARge-scale aGilE pracTices Framework
Software Service Provider
Small-scale
Large-scale
Very large-scale
Scrum
Scrum
DAD
Scrum
Component Teams x Generalized teams;
Scrum training.
Component Teams x Generalized teams;
Daily Scrum Meeting;
Planning meeting;
Wiki as Communication Tool;
Proxy Product Owner (PPO).
Integration and Unit Testing.
Daily Scrum Meeting;
Scrum of Scrums;
Product/Project Manager in Scrum;
Product Ownership;
Code freeze;
Demo presentation;
Review Meeting;
Proxy Product Owner (PPO);
Component Teams x Generalized teams;
Requirement Workshops;
Area Product Owner;
Proxy Product Owner;
Wiki as Communication Tool;
Scrum training;
Multiple Communication Modes.

Source: The author (2023)

4.8.5 Financial

Seven studies represent the financial sector involving large-scale and very large-scale projects (See table 4.8.5). The financial sector had studies involving four out of the five scaling agile frameworks present in the SLR, which are Spotify (HENRIK; ANDERS, 2012), SAFe (Leffingwell, Dean, 2023), DAD (AMBLER; LINES, 2012), LeSS (LARMAN; VODDE, 2016a), and Scrum (SCHWABER; SUTHERLAND, 2020) (See figure 4.8.5). As an exciting finding, the 13 tailored practices mapped from Spotify originated from two studies of the financial sector (SALAMEH; BASS, 2019; SALAMEH; BASS, 2020). Salameh and Bass (SALAMEH; BASS, 2019) conducted an embedded case study to evaluate the Spotify Model's applicability and agile tailoring on a large-scale B2B project of a financial company. Moreover, the same authors conducted a similar study on a Fintech company that uses the Spotify mode to discover how practitioners achieve agile tailoring using the Spotify model framework (SALAMEH; BASS, 2020).

Moreover, four studies from the financial sector used Scrum (SCHWABER; SUTHERLAND, 2020), all of them in large-scale projects (NYRUD; STRAY, 2017; LOUS et al., 2018; LEE; JUDGE; MCCRICKARD, 2011; DORAIRAJ; NOBLE; ALLAN, 2013) concentrating nine out of 32 tailored practices. Two studies on very large-scale projects were using SAFe (LAUTERT; NETO; KOZIEVITCH, 2019) and DAD (BEECHAM et al., 2021). Beecham *et al.* (BEECHAM et al., 2021) presented a study to evaluate to what degree scaling frameworks, like DAD (AMBLER; LINES, 2012), address global software development risks. Four out of nine tailored practices from DAD were built during the study evaluation. Finally, Neto and Kozievitch (LAUTERT; NETO; KOZIEVITCH, 2019) presented a survey study in which one SAFe practice was identified, the external coaches and consultants 4.4.2.

Table 11 – Financial Framework visual guide.

Market sector	Scale	Frameworks	Studies	Practices
Financial	Small-scale	N/A	N/A	N/A

	Large-scale	Spotify	S13, S22	Estimation Techniques (S13, S22) 4.3.1;
				Limited Blast Radius Technique (S13) 4.7.32;
				Support/Maintenance Squads (S13, S22) 4.3.3;
				Roadmap (S13, S22) 4.3.4;
				Establish a clear vision (S13, S22) 4.3.5;
				Definition of Done (DoD) (S13) 4.3.6;
				Postmorten Documentation Process (S13) 4.3.7;
				Squad-of-Squads Meeting (S13) 4.3.11;
				Product Owners weekly meeting (S13) 4.3.12;
				Transparency (S13) 4.3.13;
				Measurement Indicators (S22) 4.3.8;
				Architectural Decision Process (S22) 4.3.9;
				Knowledge Sharing Process (S22) 4.3.10.

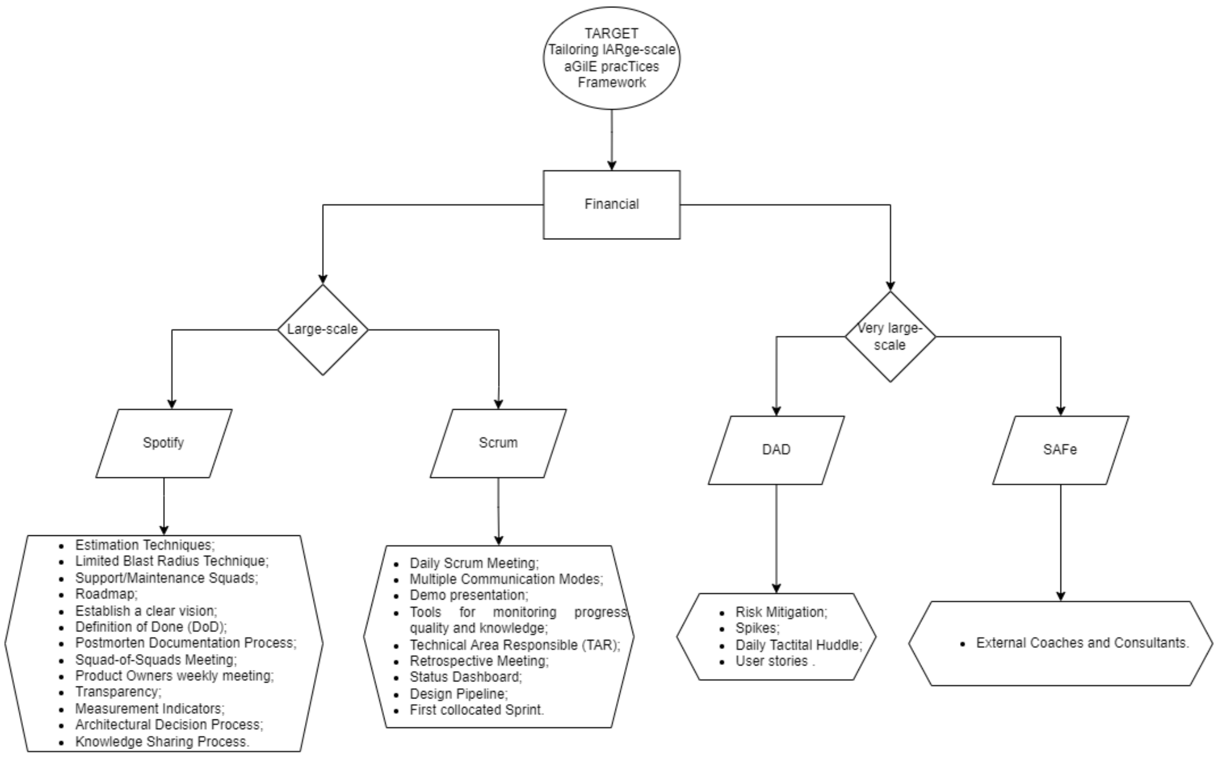
		Scrum	S19, S25, S34, S68	Daily Scrum Meeting (S19, S34) 4.7.1;
				Multiple Communication Modes (S19, S34) 4.9.2.9;
				Demo presentation (S19) 4.7.8;
				Tools for monitoring progress, quality and knowledge (S19) 4.7.12;
				Technical Area Responsible (TAR) (S19) 4.7.22;
				Retrospective Meeting (S25) 4.7.3;
				Status Dashboard (S34) 4.7.4;
				Design Pipeline (S34) 4.7.29;
				First collocated Sprint (S68) 4.7.11.

	Very large-scale	DAD	S61	Risk Mitigation (S61) 4.5.1;
				Spikes (S61) 4.5.2;
				Daily Tactital Huddle (S61) 4.5.4;
				User stories (S61) 4.5.5.

		SAFe	S50	External Coaches and Consultants (S50) 4.4.2.

Source: The author (2023)

Figure 14 – Financial Framework visual guide.


TARGET Tailoring lARge-scale aGilE pracTices Framework
Financial
Large-scale
Very large-scale
Spotify
Scrum
DAD
SAFe
Estimation Techniques; Limited Blast Radius Technique; Support/Maintenance Squads; Roadmap; Establish a clear vision; Definition of Done (DoD); Postmorten Documentation Process; Squad-of-Squads Meeting; Product Owners weekly meeting; Transparency; Measurement Indicators; Architectural Decision Process; Knowledge Sharing Process.
Daily Scrum Meeting; Multiple Communication Modes; Demo presentation; Tools for monitoring progress quality and knowledge; Technical Area Responsible (TAR); Retrospective Meeting; Status Dashboard; Design Pipeline; First collocated Sprint.
Risk Mitigation; Spikes; Daily Tactital Huddle; User stories .
External Coaches and Consultants.

Source: The author (2023)

4.8.6 Process & Industry Automation

Four studies using Scrum represent the Process and Industry Automation sector. Most of them involve large-scale projects (BASS, 2015; VÄLIMÄKI; KÄÄRIÄINEN, 2008; SABLIS; SMITE; MOE, 2021), and only one from a very large-scale scenario (DANEVA et al., 2013) (See table 4.8.6). Only the daily Scrum meeting 4.7.1 is presented in two of the studies (BASS, 2015; VÄLIMÄKI; KÄÄRIÄINEN, 2008). Meanwhile, the other practices from the sector are presented only in one study per time (See figure 4.8.6).

The four studies conducted the company's projects in two continents: Europe and Asia.

Table 12 – Tailored practices from the Process and Industry Automation sector.

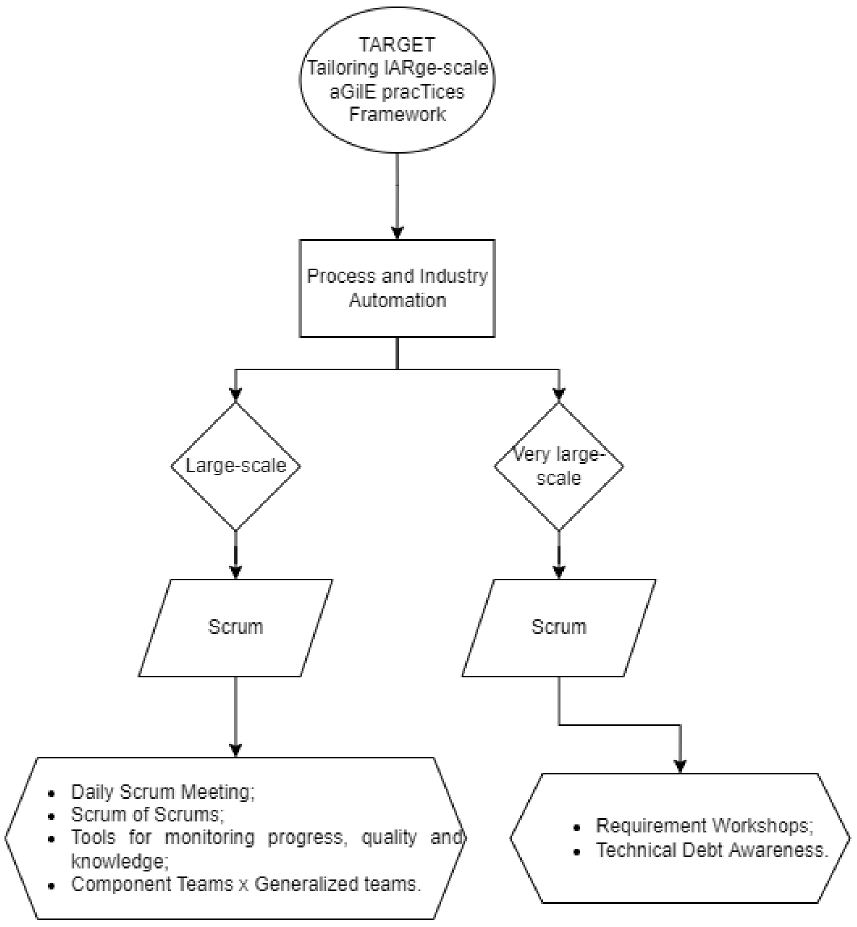
Market sector	Scale	Frameworks	Studies	Practices
Process & Industry Automation	Small-scale	N/A	N/A	N/A

	Large-scale	Scrum	S41, S55, S60	Daily Scrum Meeting (S41, S55) 4.7.1;
				Scrum of Scrums (S55) 4.7.2;
				Tools for monitoring progress, quality and knowledge (S55) 4.7.12;
				Component Teams x Generalized teams (S60) 4.7.15.

	Very large-scale	Scrum	S67	Requirement Workshops (S67) 4.7.17; Technical Debt Awareness (S67) 4.7.19.

Source: The author (2023)

Figure 15 – Process and Industry Automation Framework visual guide.


TARGET
Tailoring lARge-scale aGilE pracTices Framework
Process and Industry Automation
Large-scale
Very large-scale
Scrum
Scrum
Daily Scrum Meeting;
Scrum of Scrums;
Tools for monitoring progress, quality and knowledge;
Component Teams x Generalized teams.
Requirement Workshops;
Technical Debt Awareness.

Source: The author (2023)

4.8.7 Internet

Similar to the Enterprise CRM sector, the Internet sector originates itself through the studies of Bass (BASS, 2014; BASS, 2013; BASS, 2012; BASS, 2015) that evaluated the Scrum Master activities at large enterprise projects (BASS, 2014), product owner functions and teams in distributed projects (BASS, 2013; BASS, 2015), and also the influences on agile tailoring at enterprise software development (BASS, 2012). In all of the studies, the same company was evaluated, and it involved the development of products on the internet related to mail, calendar, and options.

The company in the multiple-case studies had projects spread across India and USA, with many teams configuring very large-scale projects. The Internet company used tailored Scrum in all studies, with 9 out of 32 practices mapped from Scrum (See table 4.8.7 and see figure 4.8.7).

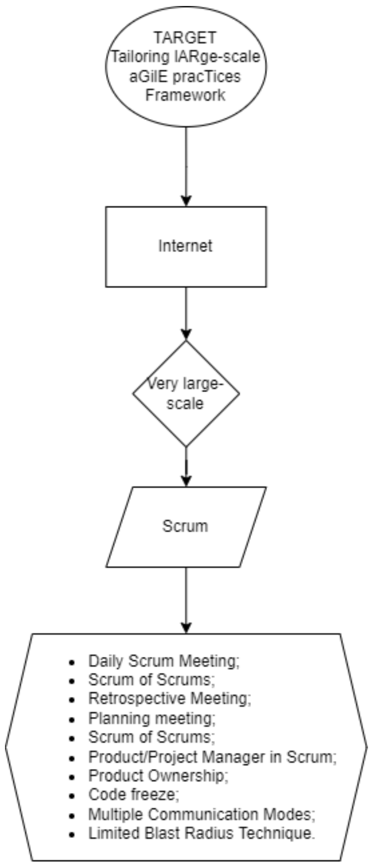
Table 13 – Tailored practices from the Internet sector.

Market sector	Scale	Frameworks	Studies	Practices
Internet	Small-scale	N/A	N/A	N/A
	Large-scale	N/A	N/A	N/A

	Very large-scale	Scrum	S3, S5, S8, S41	Daily Scrum Meeting (S3, S41) 4.7.1;
				Scrum of Scrums (S3, S5) 4.7.2;
				Retrospective Meeting (S3) 4.7.3;
				Planning meeting (S3, S41) 4.7.5;
				Product/Project Manager in Scrum (S5) 4.7.7;
				Product Ownership (S5, S41) 4.7.16.
				Code freeze (S8) 4.7.24;
				Multiple Communication Modes (S8) 4.7.6;
				Limited Blast Radius Technique (S8) 4.7.32.

Source: The author (2023)

Figure 16 – Internet Framework visual guide.



Source: The author (2023)

4.8.8 Oil and Energy

Three studies represent the Oil and Energy sector. All of them are related to developing and maintaining products to control power, energy, and oil refinery systems (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA;

DURASIEWICZ; LASSENIUS, 2009a) (See table 4.8.8). Both Paasivaara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a) studies and Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011) case describe the implementation of Scrum tailored practices in large-scale projects from the oil and energy sector (See figure 4.8.8).

The three studies are classified as large-scale projects. Paasivara's *et al.* studies involved seven teams spread across Norway and Malaysia (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a), while Hossain *et al.* study involved two teams spread across Finland and an offshore country (HOSSAIN; BANNERMAN; JEFFERY, 2011). A total of nine out of 32 Scrum practices are presented in the three studies. Meanwhile, only the Daily Scrum meeting practice 4.7.1 was applied in both studies.

Table 14 – Tailored practices from the Oil and Energy sector.

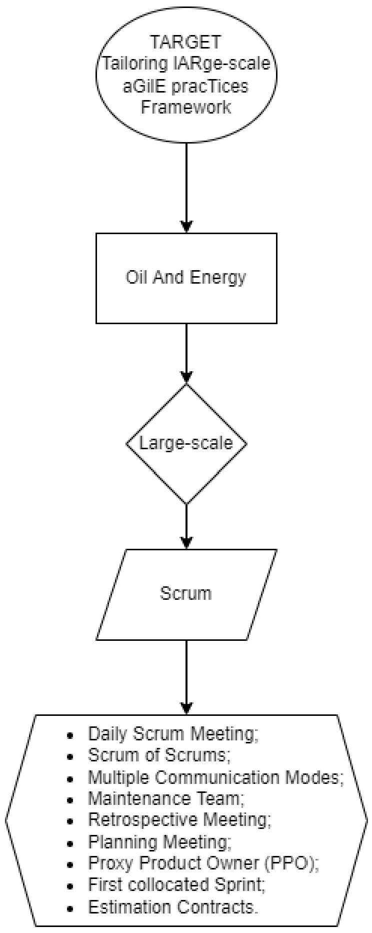
Market sector	Scale	Frameworks	Studies	Practices
Oil and Energy	Small-scale	N/A	N/A	N/A

	Large-scale	Scrum	S11, S23, S59	Daily Scrum Meeting (S11, S23, S59) 4.7.1;
				Scrum of Scrums (S11, S59) 4.7.2;
				Multiple Communication Modes (S11) 4.7.6;
				Maintenance Team (S11, S59) 4.7.21;
				Retrospective Meeting (S23, S59) 4.7.3;
				Planning Meeting (S23, S59) 4.7.5;
				Proxy Product Owner (PPO) (S23) 4.7.10;
				First collocated Sprint (S59) 4.7.11;
				Estimation Contracts (S59) 4.7.23.

	Very large-scale	N/A	N/A	N/A

Source: The author (2023)

Figure 17 – Oil and Energy Framework visual guide.



Source: The author (2023)

4.8.9 Enterprise CRM

The enterprise CRM studies concentrate on the companies presented in several studies from Bass (BASS, 2013; BASS, 2012; BASS, 2015). In those studies, Bass investigated several questions regarding product owner functions and teams (BASS, 2013; BASS, 2015), and also the influences on agile tailoring at enterprise software development (BASS, 2012). During the multiple-case studies, three companies specialized in CRM were evaluated. Those companies developed enterprise CRMs for several domains, such as insurance, banking, healthcare, credit card, financial services, and core business (BASS, 2013; BASS, 2012; BASS, 2015).

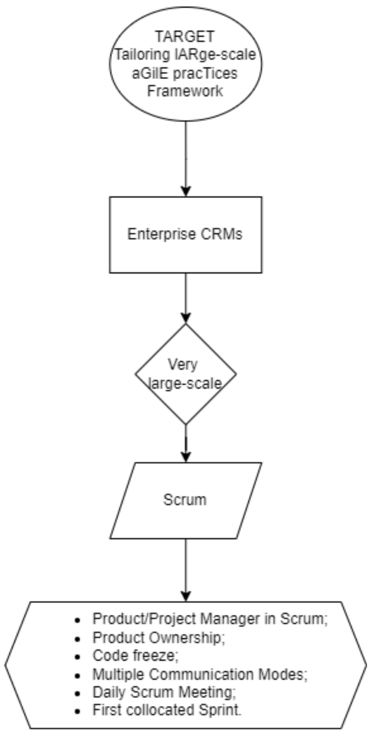
All of those case studies involved CRM companies tailoring Scrum in large-scale environments (See table 4.8.9 and see figure 4.8.9). Moreover, the companies used six out of 32 practices from Scrum.

Table 15 – Tailored practices from the Enterprise CRM sector.

Market sector	Scale	Frameworks	Studies	Practices
Enterprise CRM	Small-scale	N/A	N/A	N/A
	Large-scale	N/A	N/A	N/A
	Very large-scale	Scrum	S5, S8, S41	Product/Project Manager in Scrum (S5) 4.7.7; Product Ownership (S5, S41) 4.7.16; Code freeze (S8) 4.7.24; Multiple Communication Modes (S8) 4.7.6; Daily Scrum Meeting (S41) 4.7.1; First collocated Sprint (S41) 4.7.11.

Source: The author (2023)

Figure 18 – Enterprise CRM Framework visual guide.



Source: The author (2023)

4.8.10 Automotive Industry

Only two studies represent the automotive industry. However, one of them presents an exciting environment of a very large-scale automotive industry company that was adopting LeSS in four different products (ULUDAG et al., 2019) (See table 4.8.10 and see figure 4.8.10). The four products are software systems that are part of the chain of the automotive company, and together, the products involve more than 800 employees spread across five countries (ULUDAG et al., 2019). In the study, the authors presented how the different products adopted, applied, and tailored the LeSS framework in different scenarios. Seven out of 17 practices from LeSS are present in the study from Uludag *et al.* (ULUDAG et al., 2019). Of those seven practices, four of them are just present in the study, which are Less Huge 4.6.4, inspect and adapt 4.6.6, Definition of Done (DoD) 4.6.9, and domain PO 4.6.17.

On the other side, only one Scrum practice, Product/Project manager in Scrum 4.7.7, was seen from Martini *et al.* (MARTINI; PARETO; BOSCH, 2013) multiple case studies involving an automotive company in a large-scale project.

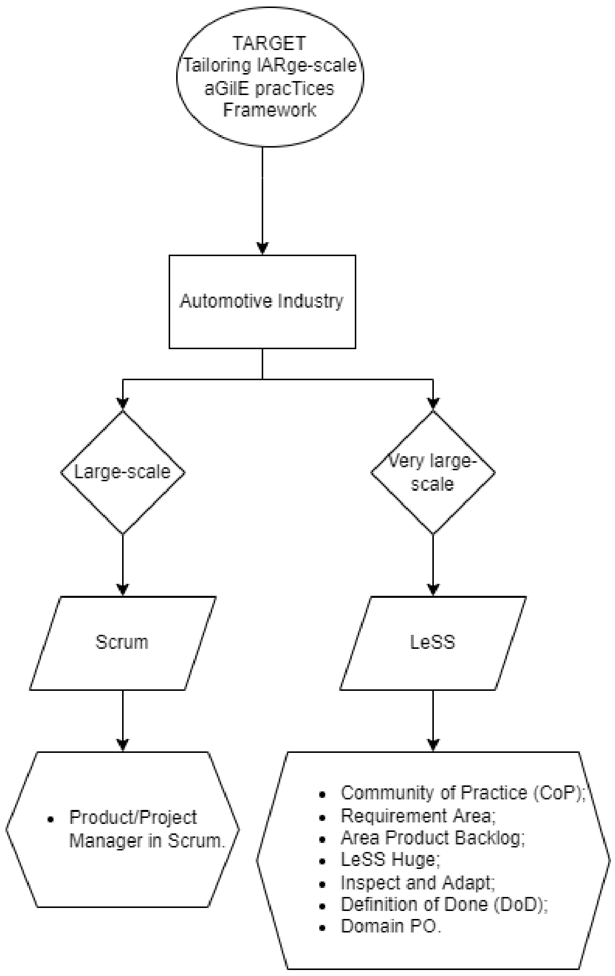
Table 16 – Tailored practices from the Automotive Industry sector.

Market sector	Scale	Frameworks	Studies	Practices
Automotive Industry	Small-scale	N/A	N/A	N/A
	Large-scale	Scrum	S17	Product/Project Manager in Scrum (S17) 4.7.7.

	Very large-scale	LeSS	S1	Community of Practice (CoP) (S1) 4.6.1;
				Requirement Area (S1) 4.6.2;
				Area Product Backlog (S1) 4.6.3;
				LeSS Huge (S1) 4.6.4;
				Inspect and Adapt (S1) 4.6.6;
				Definition of Done (DoD) (S1) 4.6.9;
				Domain PO (S1) 4.6.17.

Source: The author (2023)

Figure 19 – Automotive Industry Framework visual guide.



Source: The author (2023)

4.8.11 Healthcare

In the healthcare sector, one study from Gupta *et al.* (GUPTA; VENKATACHALAPATHY; JEBERLA, 2019) has shown the use of tailoring practices from both SAFe and Scrum in the same project while implementing the DevOps approach. In this study, three out of 24 practices from SAFe were used, and only project increment workshop 4.4.6 was present in the study(GUPTA; VENKATACHALAPATHY; JEBERLA, 2019) (See figure 4.8.11). Meanwhile, from Scrum, the study just presented the status dashboard practice 4.7.4.

In another study from Gupta in a similar healthcare case, the case involved a very large-scale project using Scrum in a legacy project (GUPTA; JAIN; SINGH, 2018). In the project, the authors reported the presence of six out of 32 Scrum practices (See table 4.8.11).

Table 17 – Tailored practices from the Healthcare sector.

Market sector	Scale	Frameworks	Studies	Practices
Healthcare	Small-scale	N/A	N/A	N/A

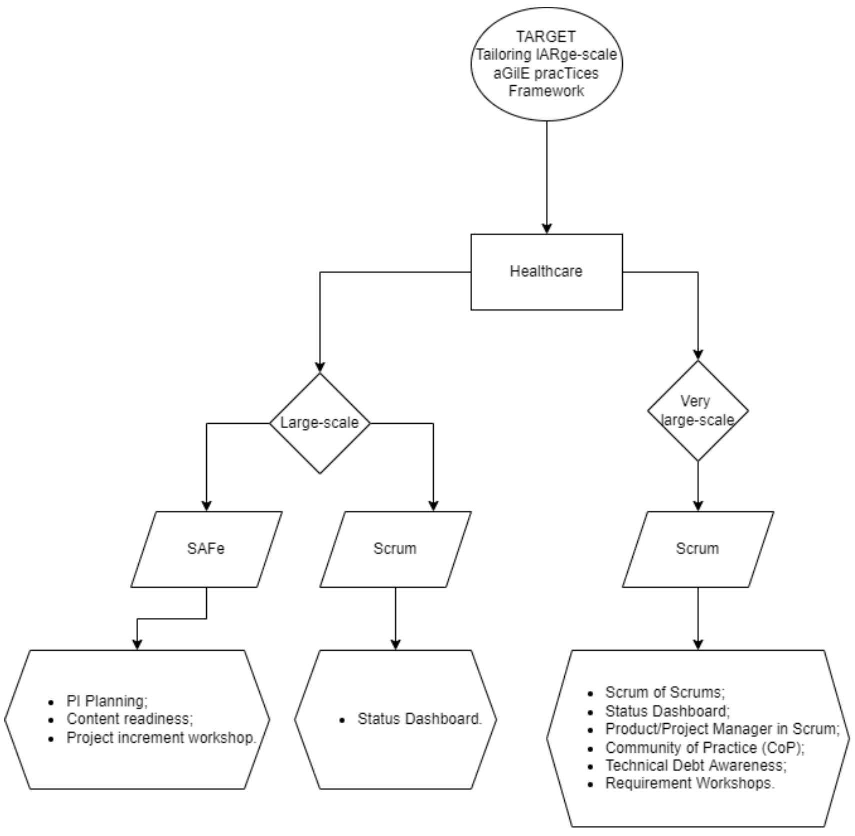
	Large-scale	SAFe	S33	PI Planning (S33) 4.4.1;
				Content readiness (S33) 4.4.3;
				Project increment workshop (S33) 4.4.6.

		Scrum	S33	Status Dashboard (S33) 4.7.4.

	Very large-scale	Scrum	S31	Scrum of Scrums (S31) 4.7.2;
				Status Dashboard (S31) 4.7.4;
				Product/Project Manager in Scrum (S31) 4.7.7;
				Community of Practice (CoP) (S31) 4.7.25;
				Technical Debt Awareness (S31) 4.7.19;
				Requirement Workshops (S31) 4.7.17.

Source: The author (2023)

Figure 20 – Healthcare Framework visual guide.


TARGET
Tailoring lARge-scale aGilE pracTices Framework
Healthcare
Large-scale
Very large-scale
SAFe
Scrum
Scrum
PI Planning;
Content readiness;
Project increment workshop.
Status Dashboard.
Scrum of Scrums;
Status Dashboard;
Product/Project Manager in Scrum;
Community of Practice (CoP);
Technical Debt Awareness;
Requirement Workshops.

Source: The author (2023)

4.8.12 Optical Industry

Only two studies were held in the optical Industry sector, specifically in the same company with headquarters in Dublin but with teams across North America (See table 4.8.12). However, in the study from Razzak *et al.* (RAZZAK et al., 2018) in 2018, the company was classified as large-scale since it presented less than ten teams. Meanwhile, in Beecham *et al.* (BEECHAM et al., 2021) case study in 2021, the case had more than ten teams in the same countries, configuring it as very large-scale.

Different practices are mapped from the studies since they have different purposes. In Razzak *et al.* (RAZZAK et al., 2018), the authors aim to identify, measure, and evaluate the adoption of SAFe recommended practices across the organization, which resulted in the description of some tailoring approaches. Meanwhile, Beecham *et al.* (BEECHAM et al., 2021) presented a study to evaluate to what degree scaling frameworks, like SAFe, address global software development risks. Finally, only the Staff Members for POs' activities 4.4.4 practice was present in both studies, and a total of ten out of 24 SAFe practices are currently in the optical industry sector, which nine of them are only present in those two studies (RAZZAK et al., 2018; BEECHAM et al., 2021) (See figure 4.8.12).

Table 18 – Tailored practices from the Optical Industry sector.

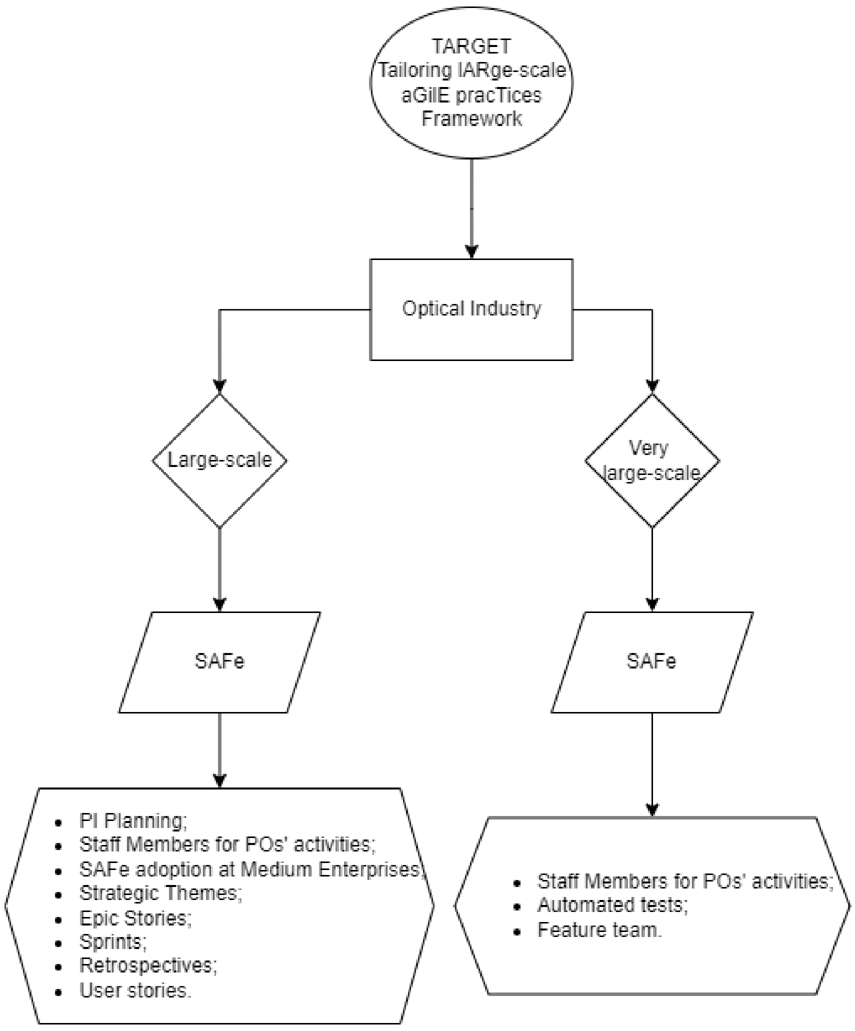
Market sector	Scale	Frameworks	Studies	Practices
Optical Industry	Small-scale	N/A	N/A	N/A

	Large-scale	SAFe	S6	PI Planning (S6) 4.4.1;
				Staff Members for POs' activities (S6) 4.4.4;
				SAFe adoption at Medium Enterprises (S6) 4.4.5;
				Strategic Themes (S6) 4.4.17;
				Epic Stories (S6) 4.4.18;
				Sprints (S6) 4.4.19;
				Retrospectives (S6) 4.4.20;
				User stories (S6) 4.4.21.

	Very Large-scale	SAFe	S61	Staff Members for POs' activities (S61) 4.4.4;
				Automated tests (S61) 4.4.11;
				Feature team (S61) 4.4.12.

Source: The author (2023)

Figure 21 – Optical Industry Framework visual guide.



Source: The author (2023)

4.8.13 Smaller Market Sectors

The market sectors within less than four practices can not be considered well representative, although it provides some relevant information regarding agile tailoring. Due to it, the following sections will describe five of those market sectors within a small portion of agile tailored practices.

4.8.13.1 Science and Research

Two studies make a part of the science and research sector. The studies were conducted in companies with teams spread in Europe and North America involving large-scale projects using Scrum (FITZGERALD et al., 2013b; GARBAJOSA; YAGÜE; GONZALEZ, 2014) (See table 4.8.13.1

and see figure 4.8.13.1). Both papers used only three practices from the 32 practices from Scrum.

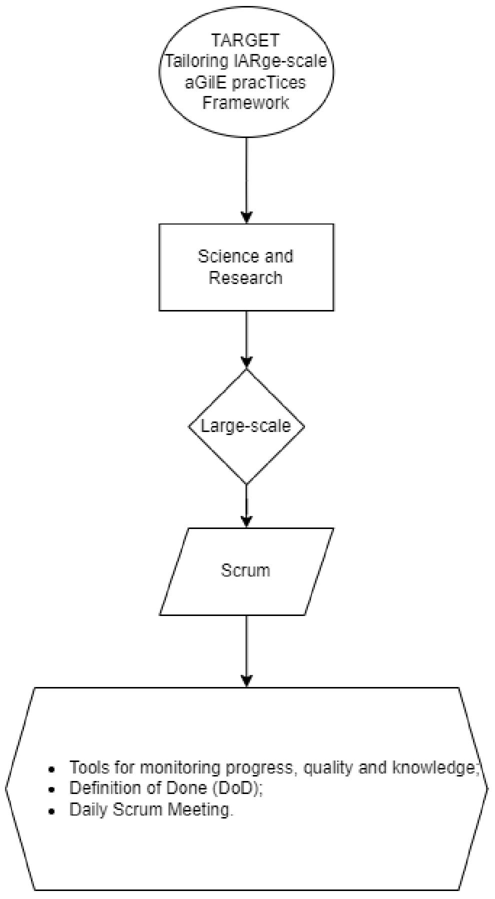
Fitzgerald *et al.* (FITZGERALD et al., 2013b) conducted the case study to identify how agile methods could be scaled to regulated environments. However, only the DoD 4.7.14 and Tools for monitoring progress, quality, and knowledge 4.7.12 practice were extracted. Meanwhile, Garbajosa *et al.* (GARBAJOSA; YAGÜE; GONZALEZ, 2014) conducted multiple case studies on projects from a research facility laboratory spread across Finland and Spain to evaluate the impact of infrastructure on the communication of agile global software development teams. During the study and only the daily scrum meeting practice was tailored 4.7.1.

Table 19 – Tailored practices from the Science and Research sector.

Market sector	Scale	Frameworks	Studies	Practices
Science and Research	Small-scale	N/A	N/A	N/A
	Large-scale	Scrum	S14, S37	Tools for monitoring progress, quality and knowledge (S14) 4.7.12; Definition of Done (DoD) (S14) 4.7.14; Daily Scrum Meeting (S37) 4.7.1.
	Very large-scale	N/A	N/A	N/A

Source: The author (2023)

Figure 22 – Science and Research Framework visual guide.


TARGET
Tailoring lARge-scale aGilE pracTices Framework
Science and Research
Large-scale
Scrum
Tools for monitoring progress, quality and knowledge;
Definition of Done (DoD);
Daily Scrum Meeting.

Source: The author (2023)

4.8.13.2 BI and Big Data

The BI and Big data sector were represented by two studies that only applied two of the most common practices from Scrum, the Daily Scrum Meeting 4.7.1 (RALPH; SHPORTUN, 2013) and the Scrum of Scrums (SoS) 4.7.2 (ROLLAND et al., 2016) (See table 4.8.13.2 and see figure 4.8.13.2). As one of the only studies in a small-scale environment, Ralph and Shportun (RALPH; SHPORTUN, 2013) presented a revelatory case study with only one Scrum team spread across two offices from America and Russia. The study investigates the Scrum abandonment of the team during the transition and the associated factors of scrum abandonment. Despite the fact that our study focused primarily on large-scale projects, such a small-scale project had one team spread between the USA and Russia, within a big timezone difference that led to similar consequences and challenges of the large-scale projects (RALPH; SHPORTUN, 2013).

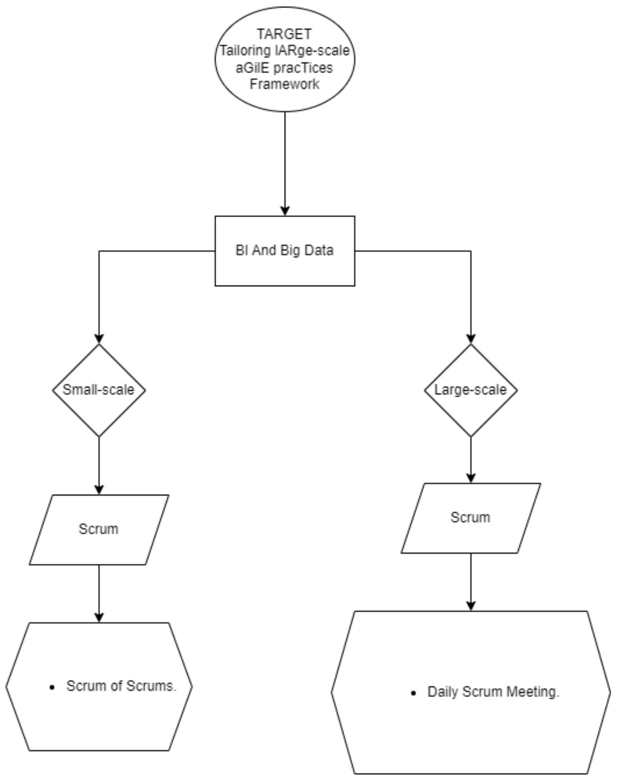
In the other study, Rolland *et al.* (ROLLAND et al., 2016) conducted a case study of a large-scale agile project that involved 120 participants in a large company from Norway. As an output of the study, the authors offer a new set of assumptions for "Agile in the large".

Table 20 – Tailored practices from the BI and Big Data sector.

Market sector	Scale	Frameworks	Studies	Practices
BI and Big Data	Small-scale	Scrum	S64	Scrum of Scrums (S64) 4.7.2.
	Large-scale	Scrum	S69	Daily Scrum Meeting (S69) 4.7.1.
	Very large-scale	N/A	N/A	N/A

Source: The author (2023)

Figure 23 – BI and Big Data Framework visual guide.



Source: The author (2023)

4.8.13.3 Logistics

Only one study represents the logistics sector in a very large-scale scenario using only the Planning meeting practice from 4.7.5 Scrum (SCHEERER; SCHIMMER; KUDE, 2014) (See table 4.8.13.3 and see figure 4.8.13.3).

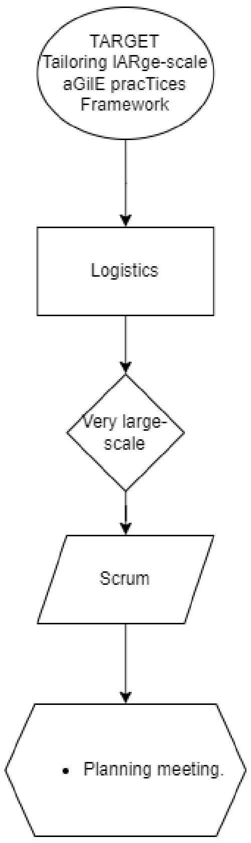
Scheerer and Kude's case study (SCHEERER; SCHIMMER; KUDE, 2014) investigated data from 125 teams that were involved in the development of a logistic management solution.

Table 21 – Tailored practices from the Logistics sector.

Market sector	Scale	Frameworks	Studies	Practices
Logistic	Small-scale	N/A	N/A	N/A
	Large-scale	N/A	N/A	N/A
	Very large-scale	Scrum	S65	Planning meeting (S65) 4.7.5.

Source: The author (2023)

Figure 24 – Logistics Framework visual guide.



Source: The author (2023)

4.8.13.4 Mission-Critical Software

Cho (CHO, 2007) presented the unique case study from our review on a distributed software development company that develops mission-critical and large-scale projects in the United States (See figure 4.8.13.4). The study had interviewees, observations, and a survey with 30 members of a large project to examine how communication, coordination, and control issues were managed and what technologies were used to mitigate the difficulties from distributed settings.

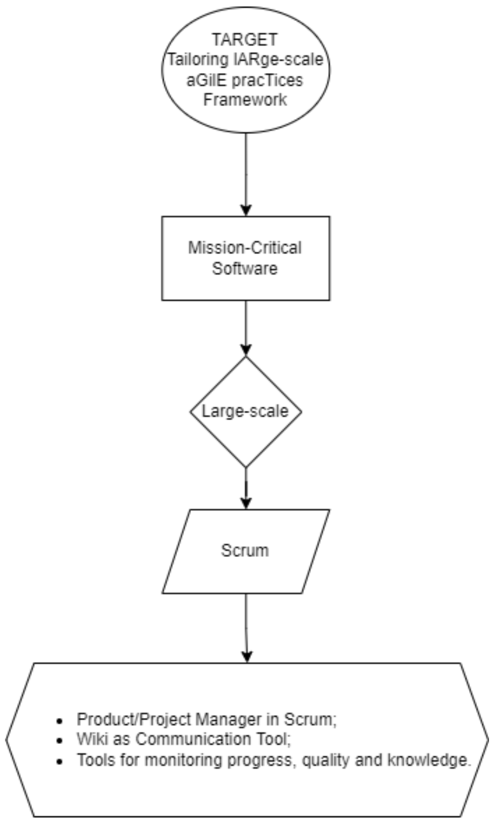
A mission-critical specialized company like the one presented by Cho (CHO, 2007) may require more traditional approaches due to the development solutions' complexity. Based on it, it was possible to see that in the Scrum practices applied, such as Product/Project Manager in Scrum 4.7.7 (CHO, 2007) (See table 4.8.13.4).

Table 22 – Tailored practices from the Mission-Critical Software sector.

Market sector	Scale	Frameworks	Studies	Practices
Mission-Critical software	Small-scale	N/A	N/A	N/A
	Large-scale	Scrum	S73	Product/Project Manager in Scrum (S73) 4.7.7; Wiki as Communication Tool (S73) 4.7.9; Tools for monitoring progress, quality and knowledge (S73) 4.7.12.
	Very large-scale	N/A	N/A	N/A

Source: The author (2023)

Figure 25 – Mission-Critical Software Framework visual guide.


TARGET
Tailoring lARge-scale aGilE pracTices Framework
Mission-Critical Software
Large-scale
Scrum
Product/Project Manager in Scrum;
Wiki as Communication Tool;
Tools for monitoring progress, quality and knowledge.

Source: The author (2023)

4.8.13.5 *Broadcasting*

The broadcasting sector is represented by just one study at one of the biggest broadcasting companies in the world involving a very large-scale project (See table 4.8.13.5 and see figure 4.8.13.5). To be more specific, Helena *et al.* (TENDEDEZ; FERRARIO; WHITTLE, 2018) conducted a case study on BBC to explore the tensions between standardization and flexibility in a very large-scale agile software development environment. The study highlights how standardization impacts flexibility in a large-scale agile environment through an ethnographic process of 17 semi-structured interviews and 54 hours of direct observations with different team members from 16 teams of the television and mobile platform BBC.

The case study involved 160 teams with around 6 to 10 members per team that used four out of 32 Scrum practices. The authors also described the use of some Kanban practices, which helped in the flexibility of teams, but also caused some problems since the customizations didn't allow the management to track all the teams with a similar approach.

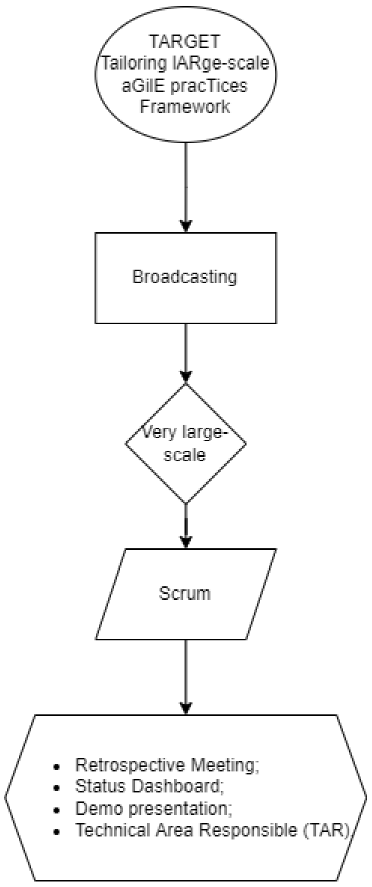
Table 23 – Tailored practices from the Broadcasting sector.

Market sector	Scale	Frameworks	Studies	Practices
Broadcasting	Small-scale	N/A	N/A	N/A
	Large-scale	N/A	N/A	N/A

	Very large-scale	Scrum	S18	Retrospective Meeting (S18) 4.7.3;
				Status Dashboard (S18) 4.7.4;
				Demo presentation (S18) 4.7.8;
				Technical Area Responsible (TAR) (S18) 4.7.22.

Source: The author (2023)

Figure 26 – Broadcasting Framework visual guide.



Source: The author (2023)

4.9 CASE STUDY RESULTS

This section will describe the results identified during the execution of the case study. The primary source of data was the transcripts from the interviews. Each of the six individuals interviewed was questioned through teleconference meetings using Microsoft Teams [5]. For data triangulation, the team members were observed during the execution of agile events through teleconference meetings. Finally, data from the project regarding product backlog, status dashboard, and development documentation were also evaluated for the case study.

During the interviews, the tailored agile practices presented in the TARGET Framework used by IT service providers from very large-scale contexts using Scrum were primarily evaluated by the team members aiming to understand whether the practices were used or not and how they were tailored. Moreover, practices from the same market sector, IT service providers but from different scale dimensions, were presented to the participants to better understand what other practices ´´outside" of their context according to the TARGET framework were also in use and how.

The results of this case study are organized as follows: a subsection to describe the team members characteristics 4.9.1, then the agile tailored practices from very large-scale Scrum IT service providers companies used by the team members 4.9.2, followed by the practices from other agile frameworks and scales dimensions from the IT Service Provider sector used by the team members 4.9.3. Finally, the considerations regarding the case study findings 4.9.4.

4.9.1 Case study demographics

The present case study gathers six interviewees. Five of them, from P0 to P4, work together in the same agile distributed team, developing a system focusing on the lead management of new students. Moreover, the P6 interviewee is the Tech leader responsible for ensuring the quality of the whole solution with a distributed team of quality. Besides developing, automating, and executing test cases of the agile distributed team from the lead management solution, the Tech leader also has to deal with the others more 20+ systems and technologies that compose the very large-scale education solution.

All of the members of this case study work in the same consultancy firm providing those development and quality services to the same client but interacting with different areas, man-

[5] www.microsoft.com/microsoft-teams

agers, leaders, and sectors.

The project in development by the client was delayed by more than two years, and the quality team managed by the Tech leader was hired to accelerate the release ensuring the minimal quality of the deliverables. The releases should be made by gathering a bunch of features representing a moment in the users' journey. The Tech leader and the QAs must ensure the quality of those moments in the release schedule. Moreover, the agile distributed team working on the lead management solution has already released the solution to production. However, the development of new features has not stopped, and team members must divide themselves between maintaining the released solution and developing a backlog of new features.

Most of the professionals from the consultancy firm working on the very large-scale solution are young, ranging from 21 to 37 years. Besides, most of them have worked for little time on the project, whereas the oldest has worked on the solution for only eight months. Regarding the interviewees' gender, two are women, and four are men.

Regarding the experience level of the interviewees, people ranged from 1,7 years to 10 years of experience. Along with it, most of the selected members have a bachelor's degree in a computing course, except P1, which comes from marketing and transitions its career. However, P0 and P5 are coursing post-graduation in software engineering.

Finally, the variety of the members selected covers most of the roles present in the project. P0 acts as a quality analyst, focusing on keeping the quality assurance philosophy alive in the project by ensuring the process is followed and not only executing functional tests. Meanwhile, P1 is the Scrum Master in the project but has been acting as a Product Owner in the last five years in other projects and is facing her first experience as an SM. P2 is the only developer specialized in the Power Automate solution from Microsoft, and consequently, Javascript due to the needed scripts to be developed. Moreover, P3 specializes in developing Microsoft Dynamics solutions that are closer to a low-code platform. P4 is the backend developer with skills in .Net, responsible for developing microservices in the project. Finally, P5 is the Tech leader of the quality team responsible for managing the team activities and the client relationship.

4.9.2 Very large-scale Scrum tailored practices

This subsection will describe the most important points that emerged from the open coding technique and the constant comparison method during the transcripts reviews regarding the tailored agile practices used by the members. A total of 15 practices from the IT Service provider sector used in very large-scale contexts using Scrum were presented to the interviewees to evaluate their practices and how. The team members used not every practice, but the majority of them. The practices are Daily Scrum meeting; Scrum of Scrums; Retrospective meeting; Demo presentation; Proxy Product Owner; Code freeze; Product/Project manager in Scrum; Product Ownership; Multiple Communication Modes; Status dashboard; Developers as Scrum Master and Product Owners; Estimation contracts; Design pipeline; Story owners; First collocated sprint.

4.9.2.1 Daily Meeting

Similar to the results seen in the Systematic Literature Review 4.1, the daily meeting was the most discussed practice among the case study practitioners. The agile distributed teams formed by P0 to P4 regularly apply a daily meeting through a teleconference solution due to the distribution of the members in the Microsoft Teams [6]. The daily usually takes 15 minutes and follows a regular template from a Scrum daily as described by a developer:

> *"The daily is held through Teams from 9:15 AM until 9:30 AM. Each member regularly passes the story they are developing to report the past progress, the planned activities for today, and any impediments.". P3, Developer.*

During the dailies, the whole dev team from the supplier is present, including the Scrum Master. On the client's side, the project coordinator is the only one always presented in the dailies according to the team (P0, P1, P2, P3, P4). Sometimes the client's Product Owner enters the meeting and the project manager just for a quick understanding of the progress since they are more worried about the entire delivery cycle of the sprint.

The daily duration of 15 minutes seems to be more a desire of the Scrum Master than the team's reality. Despite the effort of P1, it usually lasts more than 30 minutes, and the team members have a lot of perceptions of it. According to P1, the Scrum Master, since the daily meeting is the day's first meeting at 9:15 AM, people get in late and are unwilling for the meeting. From another perspective, P4 seems the daily had been used as a vent meeting. Since the product has passed from one supplier to the current distributed team, it inherited some problems from development that were only shown during the production release. Those problems are sometimes very specific to one developer and not to another, which can cause

[6] https://www.microsoft.com/microsoft-teams

the perception of a waste of time since some developers can not be involved in the discussion. Moreover, P4 has described the execution of dailies lasting 45 minutes:

> *"Even though I have more time in the project, and I've been the only one who interacted with the last supplier, the current team is kind of new to the project. Moreover, the project lived as a development project for a long time, but now it has development and maintainability tasks. Due to it, the members seem to suggest a lot of things to improve the process and reduce the impact of sprint delays, which leads to problems explanations and leads the daily meeting to take 40 to 50 minutes sometimes." P4, Developer*

Despite the duration problem of the daily, every member has reported a similar script followed by the Scrum Master during the daily execution. First, the daily is entirely conducted through the screen sharing of the Scrum Master computer presenting the status dashboard with the tasks in development at JIRA [7]. Each task in the columns is checked with the SM and the respective member responsible for it:

> *"During the daily, my goal is to cover each of the stories in the current sprint and present them on JIRA. For it, I share my screen with the team, and I start discussing the stories closer to completion by reading each card and asking if the responsible member requires any help, if it is stopped or not, if it has an impediment, and when this member is planning to finish it. During this process, I wrote all the points emphasized by the team to use later." P1, Scrum Master.*

An important point of the daily script that the Scrum Master emphasized is that the script is strictly followed to avoid interruptions from the client during the meeting. While the client is mostly worried about tasks with the highest priority and may want to discuss them in the 15 minutes, the SM ensures the daily script to discuss the whole sprint in the 15 minutes and doesn't deviate for the client only purposes:

> *"The project coordinator usually asks the team to stop what they are doing to focus on a specific story or a production bug. Due to it, I try to avoid those behaviors from him following the script and consequently minimizing the impacts on the delivery." P1, Scrum Master.*

[7] www.atlassian.com/software/jira

Independently of the issues encountered by the team, the daily meeting is never missed (P0, P2), even when some of the members cannot join due to internet issues:

> *"Whether a person can not join the daily due to external issues, we would ask them to report the progress, impediment, and planning for the day through chat. By the way, every day, the daily is held through teleconferences." P2, Developer.*

The team members were also asked about their perception of the daily meeting and the people's participation. The SM saw that a person from the client's commercial team was introduced to the meeting invite. Still, the SM preferred that this person not join the call since this person could not understand the agile development process and impact the team asking for features outside the sprint. From another very different perspective, P3 has reported that it seems the daily meetings on Monday are a waste of time since people are lazy on Monday morning. They don't even remember what they worked on before the weekend:

> *"When you come from a weekend that you tried to forget everything about your work, it is very difficult to remember what you did on Friday while you are in a meeting at 9:15 AM on Monday. Moreover, you usually don't have any problems to report since you didn't work on the weekend. Maybe the dailies on Monday could be later because I feel we are just showing that we have started the week working." P3, Developer.*

Beyond the execution of the regular daily at 9:15 AM involving the members P0, P1, P2, P3, and P4. Two other dailies emerged from the regular dailies' issues since the team had little time to present their progress and activities. In the regular dailies, impediments may not be discussed but just announced. In a similar process, blocker impediments that require refinement with other teams or the PO of the client also should be announced but not discussed. By doing this, the P1, SM, has established two new dailies: the blocker refinement daily and the impediments daily (P1).

Since a member reports a problem that may require the PO explanation or a discussion with another team from the client, another daily is called right after the regular daily at 9:30 AM. On this blocker refinement daily, the SM gathers the project coordinator and the PO from the client and also the developer involved in the problem to raise the information necessary to handle and fix such blocker:

> *"In the blocker refinement daily, I need the presence of the PO to inform about the problem and to ask him to look for a meeting with other teams to clarify the issue. I have created this process to push the PO to be more involved in the daily discussions since he does not frequently participate in the regular daily." P1, Scrum Master.*

The P1 saw the daily blockers refinement as an opportunity to engage the PO in the development process since the team does not see his presence frequently. By doing this, the Scrum Master saw the need for another daily, but more likely, status check meeting called daily for impediments. The meeting is held at 2:00 PM, representing the closure of the blocker identified in the regular daily, refined at the blockers refinement daily. At the meeting, the SM asked the PO about the resolutions steps they had defined early during the impediment refinement if the PO had discussed with the other teams if is a dependency or if he had scheduled a meeting:

> *"The daily for impediments works as a Follow-up meeting that I use to ask the PO about the dependencies fix, impediments solution, or business clarification that my team needs. Both extra dailies are not always needed, but only when an impediment arises, which is quite common since the project is new and is related to many other solutions." P1, Scrum Master.*

As described by the SM, extra dailies are common, but sometimes it does not occur. It depends directly on when the impediments appear.

Looking forward to a different perspective, P5, the tech leader for Quality assurance of the whole ecosystem, participates in different dailies since his focus is on quality. The day of the quality team starts with a similar daily at 8:30 AM involving the client through a teleconference call on Microsoft Teams. However, at this call, only some members from the supplier are presented, the tech leader, manager, and operational manager. On the client's side, the managers and coordination of the product teams are presented. The daily usually takes 30 minutes, but sometimes can take more depending on the status and issues encountered during the presentation:

> *"The QA daily with the client starts with presenting the status dashboard regarding the test cases. The test cases are planned every day, and at the beginning of the daily, we show the progress of the past day. It is more a status report than a daily*

> *meeting, but we also present the planned test cases to be executed on the current day. Any big issues or bugs involving the systems that can compromise the releases are presented to the client." P5, QA Tech Leader.*

After this daily, P5 has described the execution of the other two internal QA dailies. Unlike the development team, those two dailies are needed since the team works on shifts. The project was delayed, and the client asked the consultancy firm for a 12-hour working shift to get back to the chronogram. To accomplish it, the consultancy firm divided the team into two shifts, one that starts at the beginning of the morning and finishes in the afternoon. In contrast, the other shift begins later in the morning and ends in the evening. To accommodate both shifts and to transfer the progress from one to another, P5 has established two internal QA dailies:

> *"After the daily with the client, we start the first internal QA daily involving all the quality analysts. At this meeting, I describe the past progress from the last shift and the test cases that must be executed in the current day according to their priority. By doing this, the team organizes itself for the execution. During the execution, the team discovers which test cases are truly released for execution and which are not. By doing this, in 15 to 20 minutes, the first shift can start their work." P5, QA Tech leader*

After the first dailies, there are a few hours on the day that both shifts overlap. During this time, another daily with all the members are handled for knowledge transfer regarding the test cases execution. At this moment, P5 is responsible for synchronizing the work from the first-shift members with the members from the second shift:

> *"The members from the second shift already know what the team from the first shift are doing when they start their day. However, we must synchronize what the first shift members had left unfinished for the second shift members to continue. The meeting also takes 15 to 20 minutes focus on knowledge transfer. This kind of dailies was the best alternative to avoid people working a 12-hour shift every day while keeping them synchronized. " P5, QA Tech leader*

As pointed out by P5, the two internal QA dailies were needed to accommodate the shifts and provide the client's priority for the team. P5 believes that those dailies are required due to the complexity of the ecosystem, the current delay for the release, and to avoid communication issues among team members. Since most of the systems from the ecosystems are not ready and released, the Quality team requires constant communication and status checks off what test cases can be performed or not:

> *"The two dailies are needed anyway since we are working with test cases from systems that don't have all the features released and that constantly need to communicate among them. Due to it, we often find bugs or not implemented features during the test case execution. By finding it that way, we need to communicate with the teams, discuss the issues, clarify the problems to the managers, and also discuss internally what is ready or not." P5, QA Tech leader.*

Finally, independently of the issues in the project, P5 judges the daily with the client a micromanagement event required by the managers from the client side. Since the client likes to have control and command to understand what the consultants firm Quality team is working on, the dailies with them serve as status report dailies:

> *"They want a constant notion of what we are testing, and I understand it, but at the same time, it takes a lot of time from my day. They want to see constantly what is working or not, which team has failed in a test case scenario, which features were not implemented, and why. Due to it, our manager, that does not have a quality background, can not describe in full detail the technical aspects of those issues, and I need to be present to clarify it every day with them." P5, QA Tech leader.*

4.9.2.2 Scrum of Scrums (SoS)

Scrum of Scrums (SoS) was not present in the members' day-to-day activities (P0, P1, P2, P3, P4), although some of the members, after being presented to the practice, understand that it could be helpful in their context. Since their project is part of a large ecosystem with many other teams, the client could benefit from an SoS meeting to discuss dependencies across teams, blockers, and synchronized features:

> *"We don't execute a meeting with this proper objective involving the other teams from the client or other suppliers. But, in parallel, I need to discuss with other*

> *people from the project to obtain more information about the ecosystem and gain experience." P1, Scrum Master.*

Some developers considered the possible existence of such a meeting involving the client and other managers, but not specifically their agile team:

> *"From what I know, I never participated in a meeting similar to an SoS event. But I don't know for sure whether the client executes it or not. Maybe they can handle it, but outside our team." P2, Developer.*

From a different perspective, P3 has seen some similarities from the daily for impediments 4.9.2.1 with the SoS meeting, especially when conducted by the Scrum Master with the PO involving other teams:

> *"The Scrum Master usually met with the client to discuss general impediments and issues. But maybe, it can not be classified as an SoS meeting since mostly our SM is the only SM present in the meeting, and the others are managers, coordinators, and team members." P3, Developer.*

Despite the members' considerations in having or not the SoS meeting, most of them have seen the event as a possibly helpful event to their context (P0, P1, P2, P3, P4). By developing a solution with a shared development environment with another supplier from the client, the SoS meeting could help the agile team avoid long bugs investigations and help synchronize the backlog tasks with the other supplier. By doing this, the relationship between both teams could evolve, and issues would be resolved faster:

> *"Currently, we use a shared environment of the client with a team from another consultancy firm that develops another project module in the same platform. Constantly, we find bugs that were not introduced by our team but by the other supplier team, and such a meeting like the SoS could reduce the time spent during those investigations." P2, Developer.*

> *"I think the SoS meeting in our context makes total sense because we know there are a lot of legacy systems, other areas developing their projects, and we are developing ours. So, not only regarding impediments, but the SoS meeting could be helpful to understand what other teams are developing at the moment to avoid*

> *impacts on it and us. Since the system is shared between different teams, we could reduce the investigations of problems caused by others, and a meeting could help." P3, Developer.*

P2 and P3 have a strong opinions on how SoS meetings could reduce misunderstandings and put the teams on the same page, reducing the waste of effort spent on constantly investigating bugs.

On the other side, P5, the Quality Tech leader, has described the regular daily with the client as a daily SoS meeting. Since the managers and stakeholders from all the teams are present on a regular daily to understand the progress, impact, and dependencies of the test cases execution, in order to synchronize their work and thus release the necessary features to continue the tests:

> *"The description of an SoS meeting from the literature sounds very similar to what I do with the client's presence in our regular daily meeting. Basically, we share the status of the test cases with all the managers. Then, they discuss that information among themselves and organize themselves to implement what the quality team needs to continue the execution of the test cases. So, it is a kind of synchronization daily meeting" P5, QA Tech leader.*

4.9.2.3 Retrospective meeting

The retrospective meeting is a practice the teams aim to apply at the end of each Sprint. The SM has some clear goals with the retrospective meeting, to improve the development process, listen to the team members to understand their feelings and elaborate an action plan about the things that need improvement. However, the SM does not see the whole team with the same willingness:

> *"I understand that the retro goal is to improve the things you failed during the current interaction, so it is a continuous evaluation and improvement process. However, for it to happen, everybody must see its importance, and I feel that some members participate to participate." P1, Scrum Master.*

Although, despite this. The SM executes the retro strictly following a basic template of three columns with what went well, what did not go well, and what can be improved. The whole process of writing things on those columns is made collaboratively through Miro boards [8]. First, the SM separates a time for the members to write things on the columns. Then, the members should vote on the more pertinent items in the columns. After voting, the SM will discuss the most voted items, initiating with what went well since it concentrates on compliments. Then, most of the time is reserved for discussions regarding what did not go well and the actions that should be taken to avoid the same problems:

> *"I like to split the retrospective meeting into timeboxes for the teams to describe each of the items, then vote on them, and finally discuss the actions that should be taken. Similar written items are combined to form just one item, and outliers are sometimes discussed." P1, Scrum Master.*

Using Miro for collaborative writing of the retrospectives sounded good for the Scrum Master. But, P3 had reported a time when the team used a tool that did not present the member's name writing an item. The retrospective was more valuable for the team since the members could anonymously insert items in the columns without worrying about writing about a colleague's attitude:

> *"I think the other platform used by the Scrum Master was better since we can type things about the interaction without anyone knowing who is writing that. The platform helps avoid conflicts among team members when we need to reveal some attitude of a member without revealing who we are." P3, Developer.*

An important characteristic of the retrospective meeting of the agile team (P0, P1, P2, P3, P4) is that it is conducted right after the sprint review in the same afternoon:

> *"First, we execute the review with the client, and then we go into the retrospective only with the team. I tried this way to ensure the maximum participation of the team." P1, Scrum Master.*

Despite the SM's strategy to ensure the whole team's participation in the retrospective meetings, P1 has said they rarely made a retrospective involving all the members. Most of it occurs due to the need of team members to participate in other agendas with the client or work on some production bugs:

[8] www.miro.com

> *"It is difficult to gather all members even conducting the retro after the review. Whether I did a retrospective with all members, it was just once and never more. The client can constantly trigger the members for other purposes during the retro time." P1, Scrum Master.*

An interesting thing was seen during the interviews. Each member of the agile distributed team had described a different timebox for the retrospective meeting varying from one hour to three hours (P0, P1, P2, P3, P4). P0 has said it takes from one to three hours, P1 one and half hours to two hours, P2 only 40 minutes, P3 two hours, and P4 one and a half hours. Such discrepancy reveals the absence of the members during most of the reviews. However, looking at the invite made by the SM on Microsoft Teams, the meeting should take one and a half hours but sometimes goes with two hours.

The retrospective not only suffers from not having all members sometimes, but it sometimes does not occurs. Specifically when the client's company is passing through big events or big releases:

> *"We didn't conduct a retrospective in the last Sprint since the big event drowned us, and many production bugs need to be fixed in the meantime. Due to it, we skipped it in the previous interaction." P3, Developer.*

When asked about the client's perception regarding the outputs of the retrospective, the SM said the client's manager was not very concerned with the points discussed in the retrospective meeting. The manager asked the SM to send them the outputs of the meetings, and when he saw something interesting, he would point for discussion. However, the manager never did it until the interview:

> *"I talked to the manager to regularly discuss with him the results of the retrospectives, but he asked me to send the results so that he would point out whether finding something interesting. So, he ignores the content, but I keep sending it." P1, Scrum Master.*

Some members believed that the team implemented half of the actions described and defined during the retrospectives in the next iterations (P2, P3, P4). P3 and P4 believe the whole teams need to focus and be willing to implement those actions:

"The things discussed should not stay only in discussion. We need to execute them to see things improving. We, the team, are responsible for 90

"The team lacks focus after the retrospective to implement the actions previously discussed. Most of the time, the action plan is forgotten over time." P4, Developer.

In the quality team of P5, the retrospective is conducted more simply and is very different from the Scrum agile retrospective. The client organizes it, involves all the managers, directors, even VPs, and takes four hours, and is performed every 15 days. However, the QA Tech leader participates only for the first 30 minutes. During this time, the Tech leader is supposed to present the general evaluation from the quality team regarding the solutions test. After it, the client discusses their points internally for improvement, and the QA Tech leader leaves the meeting :

"The client's retrospective occurs every 15 days. We have a small participation in it. I should present everything we have done in the last couple of weeks and then leave it for them to discuss. I also present some of the KPIs of the tests, and the client managers discuss them based on what I have presented. But, I left before the discussion started." P5, QA Tech leader.

Despite the client's retrospective involving P5, he also makes some not scheduled sessions to discuss internally with the team ways of improving the solution ecosystem's quality assurance process. During those sessions, the quality analysts are asked what they think can improve the process. An example of it was the introduction of new internal dailies to transfer the knowledge between the first to the second shift:

"We do some internal discussions to understand what we can improve to easier our day-to-day test executions. The second internal daily meeting was born from a discussion of the analysts that felt they were losing important information due to their lack of communication. " P5, QA Tech leader.

Finally, P5 views the internal session for process improvement as necessary with the analysts. However, he thinks the retrospective with the client's managers, directors, and VPs a waste of time since the meeting content is constantly discussed with the managers during the regular dailies. Moreover, the retrospective sounds more like a review of the work done by the directors and VPs that would discuss with their managers the progress, independently of the supplier performance.

4.9.2.4 Demo presentation

Demo presentations are commonly held during the sprint reviews of the team. For it, the status dashboard on JIRA is presented, the sprint mission is clarified, and each developed story is presented:

> *"The stories presented are already validated by the QA Analyst and the PO from the client. Based on it, each story is reviewed during the sprint review and presented to the PO and the necessary stakeholders through screen sharing on a teleconference meeting on Microsoft Teams." P1, Scrum Master.*

The Scrum Master reinforced that the demo should be conducted independently of the environment available, but it is constantly shown in the homologation servers:

> *"For me, the major goal of the demo presentation is to show the product operating perfectly as expected. And, for it, I don't care which environment it was published since it is developed and working is what we are looking for." P1, Scrum Master.*

For the demo presentation, P0, the QA analyst, is responsible for conducting most of the workflows. Since the QA analyst is responsible for testing and ensuring the correct operation of the features, he emerges as the member who most understands the stories developed in a Sprint because he tested almost everything during the iteration. However, more technical stories that require coding presentation are most of the time made by the current developer that built it:

> *"I'm the one responsible for presenting the stories workflow to the client. The team does not demonstrate the developed stories except for something more technical involving source code, performance, and integration fixes. In those cases, the developer responsible for the story complements the technical aspects." P0, QA analyst.*

P4, the backend developer, reinforces that it constantly needs to complement the P0 demos to add some information regarding the source code that does not concern the QA analyst job.

An interesting characteristic of the agile team formed by P0, P1, P2, P3, and P4 is that the demo presentation only occurs in the sprint review with the stories already validated by the QA analyst, the Scrum Master, and the client's PO. This means the client's PO already saw, tested, and validated the stories presented during the demo. The PO has a specific column on the status dashboard team, and the story can only be considered done when he validates it. Based on it, the demo presentation meeting serves more as an overview of the completed stories, but not specifically for the PO since he already validated the deliverables:

> *"For sure, the demo presentation will gather only the stories validated by the PO during the sprint course. Only, and always only before his test, I can consider the story as completed. The test is made in the homologation environment." P1, Scrum Master.*

P2 was the only developer that reported how the maintainability tasks could impact the process of demo presentation. Since the team must support the production environment, many bug fixes and investigations are required to avoid impacts on the production level. However, those tasks harm the entire development process of the sprint and also the demo presentation since the team could suffer in developing the compromised stories of the sprint:

> *"Bugs that emerge on production need to be quickly investigated and fixed by us, but most of the time, it takes the whole team's attention. Doing this, we constantly compromise the stories' development and demo presentation of them." P2, Developer.*

Finally, P5 and the quality team do not execute a proper demo presentation similar to the one in the agile development team. Instead of it, P5 and his team daily run assisted test cases with different teams and managers of the ecosystem solutions. During those daily sessions, the different scenarios of the test cases are executed while the teams watch the execution through screen sharing. The main goal of those "demo assisted test execution" is to provide enough information to the teams on how the test from their systems are being conducted and how they can replicate it independently if it fails or succeeds:

> *"We execute daily assisted test execution with teams to resolve pendencies involving a specific team and more than one depending on the test case integrations. Those executions reduce time, teaching the teams how to use the data and cover*

the scenarios. Moreover, it also accelerated the process of investigating possible issues since everybody is watching the execution, not only the QA team." P5, QA Tech Lead.

4.9.2.5 Proxy Product Owner

There is no role of the Proxy Product Owner in the agile distributed team (P0, P1, P2, P3, P4). Most of the time and during the planning activities, the team relies exclusively on the client's PO for a better understanding of the business rules, and the acceptance criteria:

"The client's PO is responsible for the team's business matters; he brings the new features for developers, and we relied on him to understand what we should implement. He also clarifies the doubts regarding the business domain by describing the acceptance criteria." P0, QA Analyst.

However, despite the job executed by the client's PO. Most of the team members have seen the applicability of the PPO role on the SM, P1. The team believes that the SM naturally assumed the PPO role since the SM already played the Product Owner role in the last five years of his career (P0, P1, P2, P3, P4). By having a previous experience as a PO, the SM started pushing the team's perspective to the client's PO and manager to understand better the business domain and the technical aspects of the solution discussed by the members. Over time on the project, P1 started to gain more knowledge regarding the business domain. It helped in constant discussions with the clients about new features, bug fixes, or even the prioritization of specific stories:

"Before the presence of the SM, it was difficult for us from the team to understand and discuss constantly with the client since we barely had time to implement and test the features. However, after the SM's arrival, the discussion with the client becomes more frequent, and the SM is constantly trying to understand what the client wants, to question them, and then pass it to the team with clearance." P0, QA Analyst.

"I do not know how the SM can do this, but she can talk informally with the client and, in the meantime, extract the information we need in a much more interesting way for the implementation." P0, QA Analyst.

During the interviewers, the members also talked about how the SM was negotiating the sprint backlog with the client's PO, not only to defend the team's interests but to gather the most valuable deliverable stories for the sprint and defend it with the client's PO.

> *"I can not consider myself a total Scrum Master because I am always worried about the product's features, whether it suits the users or not. I'm also worried about ensuring the right appliance of Scrum, but I can not forget to keep pushing the team to cover all the acceptance criteria. Since we don't have a specific PO from our side, I see a need to put myself in place similar to the PPO role." P1, Scrum Master.*

Moreover, P2 has reported that the team members did not have the initiative to discuss their pain with the client, although the SM started to this represent the team, and it helped the process:

> *"P1 had discussed some of our pains with the client during our day-to-day activities. The development team does not always explicitly what we are passing to since we are constantly busy developing new things or fixing others. So, having a person do this is great. One time, we did not understand the explanation of the client's PO about a feature, and P1 realized it during the call and asked us after it. By doing this, we had another meeting to clarify the doubts." P2, Developer.*

Despite the SM acting as PPO, P1 is the most recent individual playing the SM role in the team. At least the SM has been working for only two months in the team. In the past, the team members usually acted as part-time PPOs to handle the client's request (P0, P2, P3, P4). P3 has discussed with people from the commercial team in the client organization to understand and define specific stories involving them:

> *"The project coordinator from the client has asked me to discuss the team's issues and blockers with him to find better technical solutions for the problems. Meanwhile, I have also discussed with the commercial manager some specific stories that would impact her job." P3, Developer.*

In general, some team members seems the PPO role as a need and a good solution for the problems faced in the project (P3, P4). Since the volumetry of tasks is big, a specific role in handling the team's interests and discussing technically and negotiably the project would facilitate our day-to-day work:

> *"I think it is valid to have a PPO in the team. We are constantly busy implementing new things, and a PPO could help our teammates that require more explanation about the features or even to discuss more clearly with the client about the whole solution we are building here." P3, Developer.*

The QA Tech leader has understood that he and some of the most experienced QA analysts from the team are constantly playing the PPO role. Since they execute and develop test cases involving every solution of the ecosystem, they already have a full understanding of the whole operation and integration of the systems. Due to it, P5 and the QA analysts are constantly playing the PPO role in explaining to the client their own solutions:

> *"Sometimes, we found problems and bugs during the test case execution. For example, a document sent by one system is not arriving in the other one that should receive it through an asynchronous queue. By knowing it, we can discuss the technical and business aspects with the teams from both systems to solve the issue. We basically direct them to where the error occurs, how it can be fixed, and why it is a bug of a specific system and not another." P5, QA Tech leader.*

Doing those kinds of clarifications with the client's team daily, P5 seems the PPO role is mandatory in the quality area. Since the documentation of the project does not clarify the whole points, the quality team is the one that most known about the operational and integration of the solutions:

> *"If the project had proper documentation and teams were more organized, the PPO role might not be needed. However, we don't have it, and plenty of knowledge about the proper operation of the features is on the QA team. Therefore, we must constantly act as PPOs and even POs of the development teams." P5, QA Tech leader.*

4.9.2.6 Code freeze

The code freeze practice promoted good insights with the members of the agile distributed team (P0, P1, P2, P3, P4). The team does not apply a standard code freeze policy during their development sprint. However, they managed to deal with some code freeze orders from the client that regularly established code freezes on the production server before significant events. To avoid any bugs coming from quick changes, the production server is totally frozen until the event happens:

> *"We had a big event in the last couple of weeks, and we avoided pushing new codes for the production base for three weeks. " P3, Developer.*

When asked about the frequency of those big events that would create the need for code freezing, the SM said:

> *"The client has four to six big events during the year that would require significant intervals of code freeze. However, you can see a code freeze in some repositories twice a month. " P1, Scrum Master.*

P0, the QA analyst, informed us that he handled the code freeze in the sprint by freezing the homologation environment with all the workflow he needed to test. However, it is a "gentleman" agreement between him and the other team members because any developer could break it and push code to the homologation server.

Regarding not respecting code freezes, P2 described an episode when the code freeze was not respected by one of the other teams, and it caused a significant impact on the production server. The commits were supposed to be made by 3 PM on a regular Friday, but another team member pushed new code to the production platform, which contained some bugs. Due to it, the team (P0, P1, P2, P3, P4) needed to work on a war room to find and fix the bug until late in the night, and it also impacted the next sprints that had a lot of fixes to work on:

> *"There was a week that the other consultancy firm team and we agreed on not pushing new code until 3 PM of Friday. We just agreed to avoid problems on the weekend, since we wanted to continue the work only on Monday. However, one of the guys did not know about it and pushed a lot of broken code to the platform's frontend. Our plans died, and we needed to work until late to investigate and fix the issues in the system. " P2, Developer.*

After this episode, the team members and the proper client started to value a clearer code freeze policy to avoid it from happening again. They started to plan it but are still in the ideation phase:

> *"After the huge problem in production, the code freeze is exactly what we need here. I am starting to discuss this with the SM, the other devs, and the project coordinator. We need a due date for new code pushes in the repositories. So, we have in our roadmap a task to plan those previous deployment activities and code freeze policies to follow. " P3, Developer.*

P4, the backend developer, does not believe that a code freeze would exist for the projects he works on, and he never saw anything near it on his day. Moreover, he thinks that the DevOps culture introduced by the client in the backend services, which allows pushing code at any time during the day, would be used to avoid the code freeze policies. P4 believes that the business area will not like to wait to see their features in production due to a code freeze policy applied by the IT area:

> *"When the business area requires a new feature or asks for a fix in their environment (production), they want to see it quickly available in the environment. However, code freeze policies may conflict with it and with the DevOps mindset of the organization, which defends the continuous deployment and constant deploy of new code." P4, Developer.*

Meanwhile, in the quality team of P5. The code freeze seems to be a necessity. The QA Tech leader has reported the problems he and his teams had during the data generation process. Since the test cases pass through many systems, they must build some data mass to test the features. Still, with the team constantly updating the environment, developing a data sample that worked was pretty impossible. Due to it, the code freeze started to be needed, and P5 has asked the manager of the teams:

> *"In our test cases, we need to build data samples for more than 20 systems to cover a lot of the scenarios presented on it. However, if I have a test case that covers three systems, and those three systems has 6 teams in total, I am talking about 6 teams pushing new pieces of the code hourly. Due to it, we needed to ask for a code freeze because the data mass generation was barely impossible to do and test." P5, QA Tech leader.*

Despite the need for P5, the client's organization could not accomplish a code freeze interval for all the systems due to the data sample development. However, the most important part of the ecosystem, the integration backend, which covers most of the data transactions among the systems, started to do a code freeze to help the quality team:

> *"To reduce our issues with data mass, we convince the team and the client to establish a calendar of releases in the integration backend and a code freeze policy. Since it covers most of the integrations among the solution, having a code freeze on it, we have at least a stability moment among the systems." P5, QA Tech leader.*

4.9.2.7 Product/Project manager in Scrum

The agile distributed team working on developing the client's solution has to deal with the supplier's operational managers, the client's project manager, and his project coordinator. They also considered the client's PO as a type of product manager since he knows a lot about the product, decides the new functionalities roadmap, defines them, and also validates it through the review:

> *"Besides me and the SM from the team, the client's PO is the person who masters the business domain. He also manages the expectations of the commercial areas during the features prioritization and definitions." P0, QA Analyst.*

Alongside the PO, who sometimes acts with a more traditional approach to handle the solution backlog, there is the project coordinator from the client. This one seems like a more traditional person since he constantly asks the team members to document everything they are working on:

> *"The client's project coordinator constantly interacts with us, although he is more centered on traditional processes. Everything must be documented for him, from a simple bug fix to a large feature. Once, the QA analyst found a bug related to the browser version, and the coordinator made him document the whole process, which led the QA analyst to find the issue. I don't think this is necessary, since he could work on another thing in the meantime, while a simple note of the version could be registered on JIRA." P1, Scrum Master.*

Beyond the project coordinator, there is the project manager figure, who is responsible for a business unit that covers the project the development team is working on (P0, P1, P2, P3, P4). However, such a manager does not frequently appear now. He works more outside the team strategically with other managers, but sometimes he enters the dailies or the reviews. P4 had to work closely with the project manager since he had more time and was the first person on the project. Due to it, the project manager is also a technical reference for him since they pair programs at the beginning of the project:

> *"He is the type of manager who will be closer to the operation team when he has time. He was there to develop with us when more hands programming was needed. Despite the traditional or agile thing, I appreciate this type of attitude in a manager. I had a lot of other managers in life, but the regular ones that don't get too involved in the process." P4, Developer.*

The team's perception regarding the management varied in this case. The SM does not have a good view of the project coordinator since he constantly stops the team from working on tasks that he judges as important, deviating from the sprint goals. Also, P1 sees the coordinator project as a person who is not much flexible that will defend his points independently of the teams' opinion:

> *"Once, I tried to convince the project coordinator to work with me to develop an updated project roadmap to present to the team. However, after 30 minutes, he said I would not convince him independently because he did not see any value in presenting it to the development team. That kind of behavior is closer to traditional approaches and does not please me. I think the team must participate and understand" P1, Scrum Master.*

From another perspective, P3 believes more traditional management is required since the client's organization is huge, with many sectors, teams, and business units. For P3, the high demand presented in the client's company needs a traditional process to work. Not everything could be agile in place with many users and employees to manage:

> *"The client is huge, and the agile rituals may not solve all the company's daily problems. Stop an entire sector from a business unit to participate with an agile team in the planning could be complicated. Thinking about it, I have been wondering if some traditional aspects could fit better here by providing more organization. " P3, Developer.*

P5 from the quality team described that there is no role of a project manager acting in the quality process. However, it may be supposed to have from the supplier side, but it does not. Due to it, the Tech leader frequently plays and acts as a manager to organize the team's deliverables and the test process with the client. In the P5 view, he can manage it well, and for the current time, a new person for this role can not be affordable:

> *"Most of the time, I try to organize things with the team independently. So, sometimes we test something, and if it does not go well, we fix it during the course, and that is how I deal with the absence of a real project manager. " P5, Tech leader.*

4.9.2.8 Product Ownership

Product Ownership was the type of practice in which the members had different perspectives about whether they had it or not if the whole could be considered as a team with product ownership or only some members from some roles. P0, for example, reported that seems to be much more product ownership from the client's professionals. However, he agrees that P1, the SM, also shows some product ownership while discussing with the client:

> *"I see this feeling more present in the client's PO. But, I could say that our Scrum Master also has it, especially when discussing with the client's PO, project manager, and project coordinator." P0, QA Analyst.*

However, some other members, specifically the developers P2, P3, and P4. The members have different views of product ownership. All of them described themselves as members with much product ownership. Since they constantly go beyond their typical workday by working late to fix issues, finish the development of a feature, or even investigate and fix bugs in production. Meanwhile, both developers (P2, P3, P4) agree that not everybody in the team seems to have product ownership. Still, everyone has an example of when they showed product ownership for the project:

> *"I think me and the other two developers have it. We constantly stay up late at night working on our development tasks. Sometimes we clock in and then go back to fix things and continue developing. I also think having and showing product ownership can motivate the people around us to have it too. However, having it*

> *also led us to understand some aspects of the solution that may not concern us, i.e., financial aspects." P2, Developer.*

> *"I know some people see the clock at 6 PM and basically go away. However, me and some other partners stay here until we fix what we are working on. I also think most of the people on the team have it. I think we should be proactive, not wait for things to happen, and do it frequently. One day, we found a bug before a big event, and we stayed in a war room until 10 PM to fix it." P3, Developer.*

> *"I would say I have a lot of product ownership. Since I am the only one working with the .Net microservices here. However, despite product ownership of those services, I don't see the team having it for the other things. Maybe two or three other persons have product ownership. I think we should improve our own feeling about the solution. During some episodes, it was late in the night, 9 PM to 10 PM, but the clients asked us to come in to see some bugs, and I saw other people going without being willing or not being proactive." P4, Developer.*

While P2, P3, and P4 describe their feelings of product ownership by staying late the night working on the project, although P2 has pointed out how it could impact their life quality when it becomes a routine. Making the team stay awake most of the night just for a specific deployment or a bug fix can hinder product ownership and start to demotivate the team:

> *"Once, the client needed an important deployment of a bunch of features in production involving payment methods. We kindly suffer some pressure for them to develop and deploy it quickly. However, it was not good, but we were constantly charged about the deployment that we only started at 10 PM on a Friday, and we only finished it at 5:00 AM the other day. We suffer a lot with it, and I don't think the team should feel it." P2, Developer.*

Alongside the perspectives of the agile team members (P0, P2, P3, P4). P1, the SM, has some particularities about the product ownership concept in the team. Despite the perspectives of P0 describing how P1 has product ownership, the SM itself does not see such product ownership in her. P1 recognizes herself and the team as an essential part of the process that compromises itself to aggregate value to the solution. But, it can not categorize as product ownership since the client's PO is the only one capable of changing the direction of the solution.

By the way, the SM recognizes the team's product ownership regarding the development of the features and the commitment to fix things until it is ready:

> *"I don't see myself or my role in a position of having product ownership. We, the team, are part of the development, and I am committed to frequently delivering value to it, but I don't have space for product ownership. The client's PO is the one in control. Think about it. I could have an idea that could fit the concept of promoting product ownership, but it is just an idea, and I could not interfere in any of the decisions of the product. The final word will always come from the client. I don't know if I could barely influence it if the decision is already made." P1, Scrum Master.*

Even with the team members recognizing the SM as PPO during the discussions with the client, the SM could not see itself as a person with product ownership. Maybe, for her, product ownership is more related to making decisions and not only showing commitment to the solution and its business goals.

The Tech lead perceived product ownership on the quality team as an implicit thing presented in the whole team since it is vital for the type of project. The whole seems to be very questioning about the business rules of the systems and even how they manage to communicate among them. It is common to see QA analysts asking and challenging the client's teams about how they implemented the workflows. P5 believes that behavior is vital because business rules and usability aspects would impact the testing strategies:

> *"I see many people in our team not needing my presence to question the client about a feature. I think it became very natural for the team to challenge the way the systems are presented to us, we didn't even realize it, and we had already asked the developers and managers from the client. Once, we were evaluating a test case workflow involving payments, and I did not see any feedback for the user when the client payment succeeded. I just asked the client's team about it in front of the managers and directors, and all of them realized that it was issued. For me, having product ownership in this project shows those who hired us that we are not here just to execute tests and check contracts of APIs. We are here to ensure the quality of the whole ecosystem, and it also involves asking the teams about usability, experience, and other stuff." P5, QA Tech lead.*

4.9.2.9 Multiple Communication Modes

Each member presented in this case study has reported using several technologies for asynchronous and synchronous communication (P0, P1, P2, P3, P4, P5). However, a few characteristics must be pointed out due to the context of the project.

Most of the time, the development and quality team members communicate among themselves and the client through the Microsoft Teams [9] (P0, P1, P2, P3, P4, P5). On the tool, they usually held meetings through audio and video conferences, sometimes using screen-sharing technology. Outside the sessions, every discussion is made through instant chat messages privately or in groups involving the client:

> *"I think we use Microsoft Teams 90% of our time. Since everybody is there, the communication became easier." P1, Scrum Master.*

> *"We use Microsoft Teams 24/7, it is fast, and the clients always contact us through it, so it is every time needed." P5, QA Tech leader.*

Besides Microsoft Teams, the development team reported the communication among JIRA [10] through the comments and descriptions of the development tasks present there:

> *"JIRA is also a communication tool here. Any issues, impediments, or problems are reported and described there. So, if we need to understand the status of a task without asking the responsible for it, we need to open the JIRA and look at the task." P1, Scrum Master.*

Outside those two solutions, every interviewee reported their lack of use of the email tool (P0, P1, P2, P3, P4, P5). In both the quality and development teams, email is used more by the client and supplier management team but not directly involving the team members. Due to it, the team members commonly read the emails but do not interact in the threads since they prefer the chat of Microsoft Teams.

> *"I would say that I have never sent an email in my six months here. We just use it to read, and sometimes I forget to open it." P2, Developer.*

[9] www.microsoft.com/microsoft-teams
[10] www.atlassian.com/software/jira

> *"It is very rare for us to use email. I was the only one who sometimes used it to report some of the project's artifacts to the client. By the way, the client teams use it a lot." P5, QA Tech leader.*

Finally, the members reported the informal use of Whatsapp [11] for quick messages about work and everything. The Whatsapp groups also serve as a general warning regarding meetings, delivery dates, clock-in issues, and client messages. Only the team members are present in those groups, without the involvement of anyone from the management team of both supplier and client.

> *"The groups are primarily there just for us to report some important things of the schedule if someone forgets about it. Sometimes we also talk among ourselves regarding things outside the job too." P3, Developer.*

> *"We try to use Whatsapp in the last case, but sometimes during the weekend, we need to check things in the systems and make contact through it. However, only a few people have the contact number of other people." P5, QA Tech leader.*

4.9.2.10 Status dashboard

The agile distributed team uses a digital status dashboard at JIRA [12] to manage the tasks in development, testing, and finishing (P0, P1, P2, P3, P4). Every team member has access to the digital dashboard, and the client's company purchases the JIRA licenses since every team uses Jira for development. The development dashboard represents the progress of the team in the sprint and its measurement indicators:

> *"At JIRA, we can see all the tasks the team is working on and which tasks my colleagues are finishing or having trouble with. Moreover, we also can see some KPIs of our team, such as the burndown chart." P3, Developer.*

P4 also reported another dashboard for the team: the product backlog maintained in JIRA. The client's PO is responsible for updating it, but the team is free to add any suggestion stories to it:

[11] www.whatsapp.com
[12] www.atlassian.com/software/jira

> *"There is a backlog dashboard maintained at JIRA. We can add some stories there, but the client's PO is responsible for it." P4, Developer.*

The development status dashboard used by the team has nine different lanes. The dashboard contains the tasks of a sprint for 15 days. By the time the sprint ends, the cards are archived, and when the sprint starts, the cards are moved from the backlog dashboard. The first four lanes are more critical to the developers since it represents: not started, in-progress, blocked, and developed:

> *"At the beginning of the sprint, every story is positioned on the not started lane. When the developers start their tasks, they are moved to the in-progress lane. Whether any block occurs, the task should be moved to the blocked lane, and the issue must be described on the card. Moreover, when the developer finishes a task, it should be moved to the developed lane. Finally, the card must follow the QA and deploy lanes in the developed lane." P4, Developer*

At the end of the lanes, more related to the development. The QA process start. P0, the QA analyst, is responsible for testing each task the team developed in the QA lane. When a bug occurs, he reports it to the developer and moves the task back to the developer. Meanwhile, whether a task is ok, it must follow the other lanes: QA, PO's validation, deployment at Homologation, PO's validation at Homologation, and done:

> *"When the task is developed, I bring it to the QA lane and start testing it. I would validate the tasks following the acceptance criteria defined by the client's PO. I will add evidence independently it succeeds or fails. Everything is ok, I will manage it to the PO's validation lane. The PO is supposed to make the same tests I did just to double check. Doing it, we move the story to the homologation server. Now, the developers are supposed to deploy in the homologation server. By finishing it, we start the last validation. When the PO validates the story again, but in the homologation server, whether it works on it, we move the story to the done lane. This means the developers must consider deploying it in the production server by the end of the sprint" P0, QA Analyst.*

There are a lot of steps for the development and testing of the stories during the development, although we could confirm this process with the others. Meanwhile, P4 defended that the lanes of the development dashboard and its process could be simplified. He thinks that deploy lanes could be excluded, and the quality ones could be merged:

> *"I think the lanes could be simplified. Since we have lanes for deployment and quality twice, I feel that we move the cards a lot, but without doing much or going in the right direction." P4, Developer*

Despite the lanes and the overview of the status dashboard. The members reported the type of stories used in the project. The familiar stories and bugs represent new features or issues related to a developed feature in the past that are having problems in production, respectively. Moreover, the team also works with spikes, representing a small investigation task to validate the applicability of a new feature and how it could be developed:

> *"During backlog refinement or planning, we could discuss new features requiring a spike. The spikes are used when the team does not know or does not feel comfortable implementing a feature. Due to it, the team uses the spikes to study a new story implementation. We hope the team investigates it and adds the story in the next iteration." P1, Scrum Master*

Alongside the spikes, the SM has reported the use of technical debts by the client's PO, although with a different purpose. According to the SM, when stories are not designed well, and consequently, the team implements it, the client's PO calls it technical debt when the end users ask for change. However, for the SM, the debt must be considered a business debt since the team did not originate it, and it is not technical:

> *"Sometimes, the stories were poorly designed, but the client's PO asked the team to develop them anyway. After the development, the end users asked for adjustments, and the client's PO considered it a technical debt. However, the team clarified that the design of the stories was not clear, and the issues pointed out by the end users do not relate to any technical aspects but to the business rules of the solution. By doing this, the PO continues the story without opening a new one." P1, Scrum Master.*

Despite the issues related to the story type definitions. P3 had reported that JIRA could be more explored since the tasks are not integrated into the continuous delivery pipeline, managers and developers do not notify other teams through comments in the cards, and the members do not fully report what they did in the card:

> *"We are still learning to use JIRA, but we need to go deeper. I know we could use the tool to call and reference other teams, notify a specific member regarding an impediment, or even use it as a knowledge base and integrate with the CI/CD solution of the client." P3, Developer.*

Finally, the QA Tech leader (P5) has reported a different status dashboard. Since they do not use JIRA for the test cases management, the teams use a proper tool from the supplier specifically for test management. On this tool, the concept of lanes is translated to the business units, the workflows of the solutions tested, and the risk evaluation of those workflows. In the tool, the client is supposed to follow the quality of the solutions based on its workflows, from what is more critical to less critical:

> *"First, we report the test cases through our solution for quality management, but the client has asked for new integrated views of the dashboard, and we asked the product team to work on it. During this time, we also used Excel sheets with the progress of the test cases according to the release schedule." P5, QA Tech leader.*

The Excel sheets [13] were used while the product team from the supplier company was developing the integrated dashboard views asked by the client.

4.9.2.11 Developers as Scrum Master and Product Owners

The developers of the team and the QA analyst reported that their roles are well-defined for now (P0, P2, P3, P4). However, such a thing was not always true during the whole project. The developers had described some episodes when they needed to act as SMs or POs due to the absence of those roles in the team. The beginning of the project was chaotic due to the change of supplier in the project, with the team members being the new ones:

> *"In the beginning, the client was trying to manage our arrival and the coordination from our company (supplier), but it was confusing, and we tried to manage our job and tasks together." P3, Developer.*

[13] www.microsoft.com/microsoft-365/excel

P4 also described that he thinks a developer could not be simplified as a person who codes but must look forward to helping the team in other areas, such as agility, planning, and even validations with the client. P4 is the only one responsible for the .Net services of the team. Due to it, he had to act constantly as PO and SM to discuss with other teams regarding integrations, issues, and dependencies:

> *"I am the reference of .Net on the team and the oldest squad member. Based on it, I needed to investigate and look for other teams in the client environment from the beginning to solve my implementation issues. More than this, I needed to discuss with some other managers regarding dependencies, and I negotiated with them some implementations." P4, Developer.*

Moreover, P2 understands that everybody played other roles, such as SM and PO, for a while. Especially when P1, the SM, was not in the project :

> *"I think everybody gathered to help each other understand the stories properly, discuss whether it was needed for the solution, or if the priority was right. Once, we discussed with the PO about the priority of a story to the business, but we were not heard properly." P2, Developer.*

P2 was the only member who reported that the clients were not considering what the members were discussing outside their regular activities.

The SM also pointed out that some of her activities seem to be a help for the client's PO, but it could not configure the role of a PO:

> *"I could see myself as a helper when I act with the client's PO. However, I don't have the power to add new stories to the backlog or change the priorities, but I am constantly discussing conflicts among the stories with him and the other teams. By doing this, I can not assume that I act as the PO sometimes. It would be too much." P1, Scrum Master.*

Despite the SM considerations. One member percevid her presence as a relief since he and the other developers do not need to act constantly in solving dependencies and conflicts and asking the client's PO for clarification or issues related to integrations:

> *"I saw P1 aware when we don't understand a story and ask the PO to clarify it. Also, we do not need to struggle to solve conflicts since the SM can contact the other teams first and discuss the issues with us." P2, Developer.*

In the quality team, the situation is a little bit different. The QA Tech leader perceives that the QA analysts must act as PO and SM due to the project's complexity and the absence of the roles in the client. P5 also described how he prepared three members of the team to assume the protagonism of those roles:

> *"I needed to raise some of those roles aspects in the QA analysts since I could not be present in every meeting call of the project. When I arrived, I did not perceive it on the team. Currently, I could say I have three members on the team able to discuss with the client regarding the business goals, suggest prioritization of test cases, and even question the other teams about the implementation of the features." P5, QA Tech leader.*

P5 believes that it is a matter of necessity to coach those members to play the SM and PO role. The project could not depend on him to manage all the stuff:

> *"It is impossible for me to be present 24/7 in the project, in its discussions and meetings. I realize that it will not be the same because most of those QA analysts have little experience in management. However, I do what I can with the resources I have. In our case, we do not even split the functions. The other analysts and I must be the PO or the SM when needed." P5, QA Tech leader.*

4.9.2.12 Estimation contracts

The agile distributed team did not implement the estimation contract (P0, P1, P2, P3, P4). Instead of working with estimation contracts based on the sprint estimates of effort and time, the team is billed through a timesheet. It means that the supplier had hired the team members for its company and then allocated the professional to the client project. During the day, the members are supposed to clock in at the beginning of the day, at lunch, and when they finish the workday. At the end of the month, a sheet gathering the time spent by the members working on the client is shared with the client.

By doing this, the client pays for the team members' work hours each month. The benefits of it, whether the client asks for the change of an individual, all the taxes and costs of the replacement will be held by the supplier.

In the meantime, the quality team represented by P5 is the one who works with an estimation contract. But, the team does not have a sprint, and each sprint has an estimation and a contract, as seen in some studies in the literature 4.7.23. In this case study, the estimation contract is represented by a large scope of test cases grouped at different moments following a release schedule. During the project's lifetime, the release plan changed and was delayed, some systems were not finished, and the scope of the test cases was not even ready for the teams to test. Due to it, renegotiating the scope will be necessary to accommodate the project's costs and avoid impacts on the supplier and client.

4.9.2.13 Design pipeline

The design pipeline was not present in the agile distributed team (P0, P1, P2, P3, P4). However, the members recognized the practice and had used it in past experiences (P0, P4). Due to the project technologies, all the members agreed that the design pipeline would not be necessary for the context since the project is developed based on Microsoft Dynamics 365 [14], which is a low-code platform that already has pre-defined design screen views:

> *"We don't need to have the design pipeline. Dynamics 365 delivers everything practically done. We just need to build the new forms by adding pre-defined fields to the form interface, the design of the solution follows the Microsoft design, and we do not need to change it. Being like that makes us not require a design team to build the prototypes of the forms. We can do it already by ourselves, and the design pipeline does not make sense." P3, Developer.*

P3 also describes that the developers only want to know the fields necessary to develop a new form during the planning. The presence of a prototype is irrelevant for P3, although P3 reported that the lack of a full description of the fields could impact the understatement of the story:

[14] dynamics.microsoft.com

> *"I constantly ask the PO about new forms' fields and a story's documentation. If I don't know the details of a form, I cannot implement it. But I don't need its design, just the description." P3, Developer.*

Despite the possible need for a design pipeline in the agile development teams. A quality assurance team would not need it by default, but P5 has described what he called the BDD pipeline. Since the systems are still in development, many test cases are still being built based on the release schedule. The quality team conducted the assisted journey meetings with the development teams. At those meetings, the quality team browser through the workflows of the solutions in the role of a user. Some analysts are supposed to build the BDD during the teleconference meetings with the client teams' describing the system's expected behavior during the journey execution. Such meeting occurs with new features being ready or primarily ready. Meanwhile, the assisted test meetings with other analysts used those BDDs to execute the test cases:

> *"We do what I can call a BDD pipeline. The QA analysts must absorb the full knowledge during the assisted journeys meetings with the client and the teams. The upcoming features are previously evaluated in those meetings, and the evaluation consists of building the BDD of the test cases. Further, other analysts are supposed to review those BDDs and test the cases when the features are truly done and published in the right environment." P5, QA Tech leader.*

The main goal of P5 was to allow every quality team member to participate in the assisted journey meetings with the client. It would allow everyone to know enough about the solutions and how they were implemented. However, it is not possible since the team is split for the development of BDDs and the execution of test cases:

> *"I want to make everybody pass the process of building the BDDs and executing the test cases. However, I can not do that because we need to parallelize the BDD development and the test executions. Doing that way, the QA analysts constantly need to watch recorded meetings to get along with the business rules of the new features." P5, QA Tech leader.*

Through the BDD pipeline, P5 could accelerate the quality process, but since the members that build the BDDs and execute the test cases are from the same role, they will need to synchronize through recordings to understand the workflows. Several BDDs could be more challenging to understand than a prototype workflow.

4.9.2.14 Story owners

Until the moment of the interviews, the agile members do not see themselves or anyone on the project as story owners, but they have their justifications for it (P0, P1, P2, P3, P4). P3 reinforced that the team has many contacts with the PO regarding the stories, but in the past couple of months, P3 is crossing the PO barrier and talking with some of the users and stakeholders:

> *"Our primary contact with the client is the PO. We constantly discuss the stories with him. However, I'm starting to cross him and go to other sides of the client to have contact with other areas related to the solution. For now, we start to do it that could make us story owners in the future." P3, Developer.*

P2 corroborates with part of the P3 speech by justifying the few time the project team has to become the story owners. Since the members only have six months on average, they are still gaining confidence with the business domain, the client, and the process. This means the team lacks the necessary knowledge to be story owners:

> *"I think becoming a story owner would allow us to understand the pain of the users and the client. It would also help separate the stories' owners and help the members consult each other. However, we are a young team with six months of the project, and we still need more knowledge to become it." P2, Developer.*

Beyond the necessary knowledge for it, the other members also see the story owners as a good opportunity to mature the team members as real owners of the project (P4):

> *"I think this practice is pretty cool to mature the team members and to allow them to get more involved with the project by giving more responsibility through stories. The developers or anyone as story owners can assume more protagonism by defining and defending a story." P4, Developer.*

Meanwhile, the QA described the team members as bug owners. Since a bug appears on production or even during the sprint, the team member responsible for the technology where the bug happened or the one who investigated will be responsible for explaining it to the others:

> *"For bugs, we do not rely on the client's PO. Even when he warns us about one, the team is responsible for understanding and fixing it. Maybe it is because we may create it, but not every time." P0, QA Analyst.*

At the quality team, the QA Tech lead sees the story owners as the test case owners. Moreover, he also describes the QA analysts as owners of the business line of the solution. When the Tech leader arrived at the beginning of the project, one of the QA analysts stood out as the member with more knowledge regarding the client process. Meanwhile, with the project evolution, P5 has described that some members specialized in areas involving different systems such as billing, payment, leads, and others:

> *"During the whole project, one analyst or another stood out as the reference of the project, of specific workflows, and areas. It is a natural thing. In the beginning, we relied on one QA analyst who talked with almost every client. For now, he is focused on automating tests, while other analysts became references on test cases from sectors such as finance, billing, leads, and marketing." P5, QA Tech lead.*

4.9.2.15 First collocated sprint

First collocated sprints or even an in-person meeting never happened with the whole team. However, the members already discussed it (P0, P1, P2, P3, P4), meeting each other in the client headquarters or with the supplier officer. Meanwhile, the budget has been an issue in financing flights and accommodation for the members to gather:

> *"We have members in the South to the North of Brazil, and our client stays in Sao Paulo. I think it is not affordable for the supplier to provide and is not a priority for the client." P2 Developer.*

The members also reported that past discussions for gathering at least the individuals from Sao Paulo had happened, but nothing went beyond:

> *"We have raised this type of encounter in the past. Knowing each other in person could improve our day-to-day relationship. It also formed the team spirit, which could help during development but never goes forward." P4, Developer.*

> *"I have tried to organize an in-person planning meeting with the client and the members from Sao Paulo, but the team is spread, and the clients have specific days to go to the office. Due to it, it never went forward." P1, Scrum Master.*

For P3, in-person meetings or collocated work days may not be necessary anymore. Since gathering everybody would be impossible, the team is now working together for more than six months, and the major release of go-live is already published. P3 is the only member who does not see any more value in gathering the team at this moment:

> *"We already talk to each other every day. We kind of have some intimacy and have built a relationship. We did all of this without any formal meeting or artifact. So, we became united during the problem-solving, the deployments, and I think any in-person event now would not change our relationships for better or worse." P3, Developer.*

Finally, P5 had never met any of the other members of his team since he is one of the few members living outside Sao Paulo. However, on his own, P5 reported that he would come to Sao Paulo to meet the client, his company, and work colleagues:

> *"I am going to visit São Paulo the next month. My idea is to meet our client stakeholders and gather everybody in our office to work at least one day together. We will be able to meet each other and build a bond." P5, QA Tech leader.*

4.9.3 General Agile tailored practices

In this subsection, the other practices from the TARGET framework of the IT Service provider sector were presented for the interviewees to understand which general agile practices the members were tailoring in their context despite the scale dimension and the framework in use. Based on it, it was possible to describe how the team members used the practices of Planning meetings; Requirement workshops; Behavior Driven Development; Review meetings; Definition of Done; Technical Debt Awareness.

4.9.3.1 Planning meeting

The planning meeting was a tailored practice presented in the agile distributed team. (P0, P1, P2, P3, P4). During the planning, the team interacts with the PO to understand the development of new stories. The meeting is supposed to occur in one teleconference involving the team, the client's PO, and stakeholders. During the meeting, the stories are presented by the PO, and the team is supposed to estimate the necessary effort using story points:

> *"We do a regular planning meeting similar to what Scrum says, but it is made through Microsoft Teams. In the meeting, the PO is supposed to present the stories on his views and clarify any doubts from the team. Moreover, we will estimate the stories using story points after his presentation." P0, QA Analyst.*

It is important to point out that the team already refines the stories presented by the client's PO in previous sessions. Due to it, the PO presented the new stories and clarified the remaining doubts of the members:

> *"During the planning, we will only select the refined stories for the sprint backlog. By doing this, the PO can summarize the card, and we can estimate it." P2, Developer.*

P3 reported that the planning meeting is conducted alongside several other agile events. On the last day of the sprint, the planning of the next sprint is conducted, followed by the review and retrospective meeting:

> *"The planning meeting usually occurs on Fridays and is followed by the review and retrospective meeting." P2, Developer.*

The planning meeting had a better consensus regarding the duration of the event. Almost every member agreed it takes two and a half hours to conclude. The SM is the one responsible for sending the invitation and managing it:

> *"The planning meeting is supposed to take two and a half hours, and we are constantly achieving it." P1, Scrum Master.*

P4 reported that the client had executed some PI Planning events with the internal teams and some suppliers' professionals. However, the members of the agile distributed team were not invited:

> *"The client had a PI planning, but we did not participate. I think it would be good for us to participate. I want to be there. Maybe the budget was a problem." P4, Developer.*

In the quality team, the planning meeting perception is quite different. The QA Tech lead reported the project requires a daily planning meeting to manage the different test executions throughout the day. Since the test cases are developed every day, and each day most of them are executed by the team in two different shifts, the project requires frequent planning to avoid misunderstandings and waste of test executions. For him and the team, the dailies serve as daily planning meetings:

> *"We have a daily planning meeting during our regular dailies. It can not be weekly because we need constant synchronization regarding the new BDDs to be developed and the new test cases to be tested or re-tested. I think our dailies serve the purpose of being a planning meeting." P5, QA Tech leader.*

4.9.3.2 Requirement workshop

The requirement workshop occurs before the planning meeting, and the team conducts it to refine the new features for upcoming development previously. During the meeting, the members should break down the stories into small tasks, beginning the regular planning activities (P0, P1, P2, P3, P4). During those meetings, the whole team is present, including the client's PO and the project coordinator:

> *"The team is supposed to break and refine the necessary tasks to accomplish the stories during the requirement workshop. Such activity is done after the PO presents the upcoming features. The projector coordinator is also present to understand what will be done." P1, Scrum Master.*

> *"The requirement workshop usually occurs in the middle of the sprint, one week before the next planning meeting." P2, Developer.*

The requirement workshop did not always exist. In the beginning, the team did not have it. Without it, most of the doubts and refinement activities were supposed to happen during the planning meetings, which took a long time. But not only it, the members constantly started new sprints without fully understanding the stories. This approach harmed the sprint deliverables and also made the team members continuously contact the PO for clarifications in the middle of the iteration:

> *"In the beginning, we were working in a madness loop. We did not execute refinements in the stories, and the sprints were started without a good overview of the business rules. Only after catching a story could we go forward to understand what it means to be implemented. At this time, the contracts of the sprints were never respected. But when our SM arrived, things got better." P2, Developer.*

P1 was responsible for organizing team events to avoid delivery problems. For this, P1 focused on giving the team the necessary information to execute the job correctly.

Meanwhile, two requirement workshops meetings are handled during a sprint when needed:

> *"Sometimes, one or two days before the end of the sprint, we make a small requirement workshop to catch up with the refinement stories and update everything that is needed." P2, Developer.*

The stories' refinement and clarification also require the team members' expertise. P4 reported that during the requirement workshop meetings, the PO is supposed to do the functional refinement and explain the expected behavior of a feature. Meanwhile, P4 is responsible for describing technical aspects regarding the feature implementation:

> *"During the workshop meetings, since we do not have a tech leader in the project, I am kind of doing this by describing how specific features could be implemented technically." P4, Developer.*

Finally, P5 and the quality team do not work with the requirement workshops. However, every day they are refining their test cases.

4.9.3.3 Behavior Driven Development

The Behavior Driven Development (BDD) relies especially on the QA analysts of the agile team (P0). He is responsible for gathering the information from the PO, stakeholders, and end-users to build the BDDs. By doing this, the QA analyst gets a full overview of the stories being developed by the other members:

> *"Everything related to the quality assurance process is handled by P0. The BDD is one of those things ." P1, Scrum Master*

> *"Right after the planning meeting, when the sprint backlog is defined, I start the development of the BDDs of the stories. Based on the acceptance criteria of each story, I could start the development of the BDDs." P0, QA Analyst.*

For P0, the acceptance criteria is the critical information needed to develop the BDDs, most of which came from the client's PO.

On the other side, at P5 quality team works with BDD every day, almost every hour of their day. The team also created what they called the BDD pipeline (See subsection 4.9.2.13).

P5 and his team use the BDDs to follow up on the expected behavior of the systems during the test case execution. During the execution, when something goes differently from the BDD or the BDD is missing some important steps, evidence is registered, and the QA analyst is supposed to fix it by discussing it with the teams:

> *"Everyone in our team must know how to develop a BDD, and I know everybody can do it. However, sometimes the BDD does not follow the expected behavior. For this, as a normal procedure, we need to register pieces of evidence of the test execution and check later with the team that implemented the story to understand what happened. Sometimes we forget a step or the story changes due to restrictions. It is pretty normal in the quality area." P5, QA Tech leader.*

4.9.3.4 Review Meeting

As presented above, sprint reviews frequently occur on the agile distributed team (P0, P1, P2, P3, P4) at the end of the sprint right before the sprint retrospective 4.9.2.3. The sprint review also gathers the demo presentation activity that occurs on it 4.9.2.4.

During the reviews, the team, mostly the QA analyst, is supposed to present what they accomplished and failed to deliver. They also need to clarify the impediments faced and how they managed them:

> *"The QA analyst shares the screen to present everything working. But we also need to present what was not delivered and why we failed. Which impediment we faced, problems we encountered and how we manage it to deliver the most value."*
> *P1, Scrum Master.*

The client's PO gives feedback regarding the presentation instantly. Sometimes, more technical aspects regarding source codes need to be clarified by the developers:

> *"By the time the stories are presented to the PO, he gives feedback on whether it is ok according to the acceptance criteria defined during the planning and requirement workshops. Sometimes, we also need to discuss more technical aspects. For example, I must present the source code and explain the operation when a feature is practically developed on the backend. The QA analyst can not do it."*
> *P4, Developer.*

The sprint usually takes two hours. According to P1, everybody participating in the sprint should be presented there, and the PO, project coordinator, and sometimes the project manager. Similar to other meetings, everything is presented through teleconference meetings:

> *"The review meeting is made through Microsoft Teams, and everybody presented in the sprint planning must be there. On the client's side, we will have the PO, project coordinator, and sometimes the project manager." P1, Scrum Master.*

Further, the SM reinforces the importance of the sprint review meetings and how the QA analyst, P0, is critical for it to happen. The reviews are commonly used to provide new insights regarding the user's workflows, and past implemented features. Moreover, the QA analyst presentation reduces the distance between the end-user and the client. When the developers usually lead the meeting, a more technical perspective of their job is presented, which is not always please the client:

> *"Whether I ask a developer to present a story, they will naturally explain to me what their source code, queries, and specific services are doing with the payloads.*

> *But we do not need to hear it during the reviews. I enjoy having the QA analyst present it because he can have a broader view of the business and communicate it to the client." P1, Scrum Master.*

Finally, similar to what P5 has been doing with the planning meetings 4.9.3.1, the review meetings of the quality are simply occurring every day. It's a daily review meeting. At the end of the day, P5 has a status report meeting, similar to a review meeting with the client. At this event, he is supposed to present the number of test cases conducted, the bugs encountered, the impediments faced, and what failed. Since the test backlog changes daily, and bug appear every day, the review and the planning meeting needs to occur on a daily frequency:

> *"We have a review meeting daily. We present the bugs we encountered during the day, which test cases have passed, and the next steps. Since we have new test cases and bugs every day, we need to review the delivery daily. Whether something blocks a workflow, the review needs to point it." P5, QA Tech leader.*

4.9.3.5 Definition of Done

The Definition of Done (DoD) from the agile distributed team was defined in a document developed by P1, the Scrum Master. The DoD did not imitate itself to describe a general definition of the done concept. They are specific rules for a card to be considered based on which lane of the status dashboard it is presented. For example, stories in development by the developers need to have unit tests, pass through the acceptance criteria, be available in the QA environment, all tasks developed, and evidence of the developed feature at JIRA. Meanwhile, the QA lanes and the validation from the PO also have a proper DoD concept that involves evidence of test execution and passing the acceptance criteria. Finally, deploying lanes of production and homologation also required a set of rules that related to log and contract validation, integration tests running, and the absence of conflict with past features:

> *"When I built the definition of done, I structured it based on the lanes of our status dashboard. I want to give the team a clear view of how a story or bug is done based on its progress lanes. By doing it, we can see how an 'in development' task needs to be classified as done, what a QA validation lane requires to classify*

a story as done, and what the stories need to be completed and published to production or homologation environment." P1, Scrum Master.

Beyond the DoD, the SM also worked on the documentation of the Definition of Ready (DoR), which focuses on describing what a story or bug requires to have before it is included in a sprint. It is a checklist of items based on the type of stories describing each of them that needed to be started. For example, when required, stories must be refined with built acceptance criteria, a complete story description, a prototype, and proper documentation. For spikes, it needs to have a description and a clear goal of the investigation, and it can not impact the other tasks. Finally, bugs have some rules regarding the developer responsible for investigating it, whether it comes from a ticket or not, it needs to be presented at JIRA, a description of how to reproduce it, evidence of its existence, and evaluation of the impact into the business:

"The Definition of Ready documented next to the DoD was a way out that I used to avoid poorly stories with few descriptions to be inserted in the sprint. The DoR checklist guarantees that our team will only accept stories from the client when it fulfills all the necessary items. Moreover, it also serves when the members are going to create a spike story, and they will need to have a clear goal to accomplish and a good description of the study task. It also helps to organize the process of investigating bugs. Bugs from the support team can only be investigated with a valid ticket opened and a previous investigation from the support team." P1, Scrum Master.

The members realize that DoD differed depending on their role, and they appreciate it. P0 is focused specifically on the acceptance criteria of the stories. Meanwhile, the developers also use the acceptance criteria, but they seem more worried about the tasks needed to accomplish a story:

"My goal is to fulfill the acceptance criteria during the tests. The whole story is classified as not done if one or many criteria do not succeed." P0, QA Analyst.

"I base myself on the tasks necessary to develop the story I am working on. By the time I end it, I would test it by myself to ensure it is ready for the QA." P2, Developer.

The quality team of P5 has no formal validation for the DoD definition. However, there is a simple rule used by the QA Tech leader. The test cases require the client's formal approval to be considered done. Despite the execution succeeding, the client must approve it, and for it, the BDD is used:

> *"For me, my tests are done when the solution is ready to be delivered to the end-users. However, the client needs to approve the test cases we have already tested and succeeded in. So, it will only be done after this client's approval." P5, QA Tech leader.*

4.9.3.6 Technical Debt Awareness

The team argued that technical debt awareness is intrinsic to the team. Since the team received the project with many debts from the other suppliers, technical debts were presented in the project. Many times, the team worked exclusively on the refactoring of some of the past implementations. Due to it, the constant awareness of the members needs to be always on for the project:

> *"I think we have a good sense of technical debts. We arrived at the project with many issues from the previous supplier, and we needed to work on those debts to put the project in good shape. Due to it, I would say we stay aware all the time of new debts." P2, Developer.*

P4 reported that some technical debts are inevitable due to the fast rhythm of the project. The team needs to divide itself into support tasks and the development of new features. Due to it, some good practices stay behind, and technical debts may surge:

> *"We need to work at a very fast pace developing new features, fixing bugs, and receiving new stories from sustainability. Sometimes, the good practices and design patterns stay behind, and we need to check it as technical debt for future fixes." P4, Developer.*

The team also pointed out that there is no buffer in the sprint for technical debts. Proper debts need to become new cards and should be planned and included in the sprint backlog as any other story:

"We can push the technical debts during the planning meeting, but they will count as a story. We do not have a specific buffer for technical debts." P2, Developer.

Conversely, P5 and his quality team do not work properly with technical debts. However, the test case executions reveal the technical debts of the systems. Most of the time, when an integration does not work, it is because it is not implemented or the contract between the services was not strictly followed. P5, during the interviews, described how the quality work of his team is helping the organization teams to resolve those technical debts and align the release of the features according to the primary release schedule.

4.9.4 Case study considerations

Our case study findings showed positive results regarding the adherence to the TARGET framework tailored practices from the IT service provider sector within the team members of a genuine industry case scenario. Meanwhile, as expected, several different development and quality team particularities led the members to tailor agile practices differently, not even presented in the TARGET framework.

In general, the daily meeting of distributed teams usually needs to be held through teleconference meetings to accommodate the physical distribution of the team members 4.9.2.1 similar to what we see in TARGET studies 4.7.1. Even though the whole teams from the case were working in the same timezone, dailies involving all members were not a problem. But, since the members of the quality team of P5 worked in shifts, they used a similar approach to cases with teams spread over different continents (DORAIRAJ; NOBLE; MALIK, 2012) by conducting the dailies in the overlap hours of the members from different shifts to gather them in a single meeting. Moreover, unlike the literature that usually held dailies through chat messages due to language issues (PAASIVAARA; LASSENIUS, 2010; HOLE; MOE, 2008), the members did it only when they were facing connection issues.

The presence of separated dailies for the quality teams to discuss test execution results was present in the case and in the studies of IT Service providers (VALLON et al., 2013; VALLON. et al., 2013). Further, the frequency of more than one daily in a day was only seen in the literature due to the distribution of team members in big timezones (HOSSAIN; BANNERMAN; JEFFERY, 2011; DORAIRAJ; NOBLE; MALIK, 2012). Still, in our case study, the development sometimes required three dailies in a day due to blockers, and the quality team always conducted three dailies due to the client demand and different work shifts. Such particularity was not seen in the literature due to the development of the TARGET framework, but it sounded necessary for the teams and was tailored to accommodate their process. Finally, the lack of interest of P4 in the daily meetings usually led to specific long problems discussions that had similarities with teams from the literature that opted for reducing the frequency of the daily meetings (VALLON et al., 2014; HOSSAIN; BANNERMAN; JEFFERY, 2011).

The SoS meetings were not formally presented to the development team of P0, P1, P2, P3, and P4. However, the team saw the sometimes dailies for blocker discussion and refinements as possible as SoS meetings. But it sounds closer to an alignment meeting with other team members to discuss impediments. Nonetheless, the regular remote daily meeting of P5 involving only managers of the client side is much closer to an SoS meeting seen in the TARGET framework (BASS, 2014; BASS, 2013; VALLON et al., 2013; VALLON. et al., 2013) since it involves the managers and stakeholders from all the team to follow the tests progress, results, impediments, impact, and dependencies.

The retrospective meeting from the development team of the case has not been perceived as a well-structured ceremony since the members answered different information regarding its execution. Regardless, not all members were present at the event, and the client was not very concerned about it, which had some similarities with the literature when the lack of feedback from the interested parties discouraged the execution of retrospective meetings (HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). Further, the P5 internal and informal sessions to seek improvement with the QA analysts sound much better for a retrospective meeting approach rather than a generic meeting with high executives to report progress without discussing improvements.

The demo presentation of the development team (P0, P1, P2, P3, P4) followed a regular and common agile approach seen in other studies (ROLLAND, 2016; HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010) right after the sprint review. Although, most of the references from the TARGET framework used the demo presentation and sprint review as a feedback session after the deliverables presentation. In the meantime, the event in the case study looks like it was conducted just for the sake of it. Since the PO already evaluated the deliverables during the sprint, there is no meaning in presenting it again to him during a review meeting unless other stakeholders would evaluate it. However, it serves for the PO to give general feedback on the team's progress regarding the stories that failed or not.

The SM from the development team has been recognized as the PPO of the team. Most of this acknowledgment comes from the fact that P1 has worked as a PO in the past years of its career, and the concern with product development had always been part of P1's job. Due to it, P1 has applied some of the functions seen in PPOs of the TARGET framework studies (BASS, 2013; HOSSAIN; BANNERMAN; JEFFERY, 2011; BASS, 2015) being presented in the client's side discussing the features, reporting the progress of the team, and being intermediary roles responsible for mitigating domain complexity for the team members that constantly were in doubt about business domains specifications. Further, the absence of a PPO role in the quality team led P5 and other seniors QA to embrace the function of handling business and technical aspects of the solutions to correctly execute their job.

Both the development nor quality teams do not apply a solid policy of code freeze, but all the case members see the importance of it. At this point, a consult in the TARGET Framework could help the members by accessing studies (BASS, 2014; BASS, 2012) that applied the code freeze in very large-scale scenarios that needed to handle merge issues, data mass generation, regression tests during the development of solutions similarly big to the case study.

The presence of product and project managers in the case study's development team aligns with the TARGET framework results (BASS, 2013). The project manager from the case has similar responsibilities to the one presented in the Bass study (BASS, 2013). Within more strategic themes, the project manager was concerned with the project roadmap, involving business and technical demands for the team. Moreover, the absence of such a role from the It service company obligates P5 from the QA team to act in the management activities of the team.

The product ownership awareness of the development team was not precise. Some developers related the concept of product ownership to working long hours at night fixing issues while not staying as a lack of product ownership of other members. In contrast, the SM believed it was impossible to have product ownership whether the person could not influence or make decisions about the product. On the other side, P5 and the QAs had an adherence concept of product ownership to the TARGET framework since they could question the teams regarding business definitions and workflows. Even without a proper manager person, the development has a lot to learn from the quality team in the product ownership field.

Using multiple communication nodes in the case study highlights an interesting finding related to the email tool. While several studies have shown the prevalent presence of the email (PAASIVAARA; LASSENIUS, 2010; KORKALA; PIKKARAINEN; CONBOY, 2009; HOSSAIN; BABAR; VERNER, 2009; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b) in day-to-day activities, the case study members were not using it since the client was more responsible for it. This scenario may characterize a team without good influence to participate in the formal decision process of the team.

The digital status dashboard of the development team represents a tailored kanban board with several lanes that accommodate the team's development process, as seen in the literature (GUPTA; MANIKREDDY; ARYA, 2017; HODA et al., 2010). The digital board also helped the company see the teams' progress. Conversely, the quality team using a specific quality management tool to report the test cases, their results, and progress was not seen in the IT sector studies but only in other sectors.

Acting as SM and PO has been seen to be a constant necessity of the QA team since the absence of the roles had pushed the QAs to absorb the complexity of the domain. Similar to the case studies from the TARGET framework (HODA et al., 2010; VALLON et al., 2013; HOLE; MOE, 2008; VALLON. et al., 2013), the tech leader P5 needed to coach the other QA members to be proactive enough to handle the activities of other roles not presented in the team.

At the design pipeline practice, we can realize how the uniqueness of the team would imply the applicability of the tailored practice. In Hoda's study (HODA et al., 2010), the front-end design-intensive characteristic of the project kind of pushed the teams to establish a design pipeline to accelerate the throughput of the prototypes without overheating the developers. However, in our case study, using a low-code solution eliminates the need for such a practice since the design is pre-defined in the proper platform.

The story owner practice was not applied in the development team since the teams had little domain regarding the entire solution. However, the QA team from P5 has brought a different applicability of story owner from the quality team, the test case owner. The QAs have a holistic view of the workflows since they have worked in end-to-end tests, and due to it, those QAs constantly explain to the teams the correct behavior the solutions are supposed to have based on the test cases. By doing this, the QAs could influence teams' development process and propose fixes.

Finally, regarding the agile practices from the TARGET framework, the first collocated sprints were a practice bumped due to budget issues from both clients and the supplier side.

In the practices not mapped in the TARGET framework to the very large-scale projects of the IT service providers sector, it is essential to highlight the tailored approach used by the development team in the planning meeting and the requirement workshop. Since the development team started to refine the backlog through sprints, the planning was calmer, but in the past complex stories could arise from the PO side, harming the planning event to explain its full details. Despite the improvement, the teams continue not to participate in the PI planning event of the client, such absence of team member harm their knowledge regarding the product vision and roadmap and even compromise the estimations made but business area in the PI planning without their concern.

On the other side, the different tailored approach was exposed by the QA leader P5. According to him, the regular daily with the client can be considered a daily planning meeting because the planning of the test cases needed to be daily and not based on iterations. Such an approach was not seen in the literature for regular development and not even for QA teams.

The quality team also applied a particular practice called the BDD pipeline. Since their goal is to test the entire end-to-end workflow of systems still in development, the QAs started a process of assisted test journey meetings to gain knowledge about the business flows and develop the BDD test cases before the project release. Different from the problems perceived with BDD and the overhead with automation in the literature (VALLON et al., 2013; VALLON. et al., 2013), the QA members from the case were not automating it but developing a knowledge base of test cases about the solutions before the project went to production. Such a tailored approach was essential for the QAs members. By doing the BDD pipeline, the whole team and the client could access a living database of test cases that described the entire workflow and the expected behaviors of each of them.

Regarding the definitions of done and ready, the team usually uses the DoR technique to establish what a bug or story should have before being included in the sprint. A similar practice was seen only in Uludag *et al.* (ULUDAG et al., 2019) study with cases using LeSS. Further, the use of DoD followed a similar purpose to the studies from the TARGET framework. Considering the project's specifications, a checklist of items was developed to confirm whether a card was done.

The presented case study effectively covered several practices from the TARGET framework and reinforced the model's alignment with the industry case. Although, the most important findings rely on the newly tailored techniques described by the case's members, which give a broader view of a different context and improve the TARGET frameworks findings, and also its structure. Moreover, based on the case study results, we also evaluated the findings regarding which agile, tailored practices the team members were entirely using, partially using, and not using at all. The summary of such evaluation can be seen in the following table 4.9.4.

We believe that whether the case team has used the TARGET framework since the begin-

Table 24 – Agile Tailored practices usage

Usage	Practices

Practices fully tailored by the team members	Daily Meeting;
	Retrospective Meeting;
	Demo Presentation;
	Proxy Product Owner (PPO);
	Produc/Project Manager in Scrum
	Multiple Communication modes;
	Status Dashboard;
	Developers as Scrum Master and Product Owners.

Practices partially tailored by the members	Scrum of Scrums;
	Product Ownership;
	Story Owners.

Practices not even used by the team members	Code Freeze
	Estimation Contracts;
	Design Pipeline;
	First Collocated Sprints.

Source: The author (2023)

ning of the project, several issues faced by the members could be addressed. But, we reinforce that for it to be well applied, the team would need a manager role more present in the project since the beginning because they entered the project in the middle of it.

5 DISCUSSION

In this chapter, we aim to discuss the findings of our study. Discuss the motivation for developing the TARGET framework for the industry and academia, the best ways for the companies to use it, and how the TARGET frameworks can favor the process of agile tailoring. Finally, the threats to the validity of our study.

5.1 FINDINGS DISCUSSION

Many organizations have been transitioning their development to distributed large-scale settings. Meanwhile, transitioning from small-scale collocated agile teams to large-scale distributed teams led those organizations to have people in different locations, which led to many problems related to communication, coordination, and control of the development process (FITZGERALD et al., 2013b). Furthermore, some studies showed that 40 percent of global software development projects have failed (Betz; Makio; Stephan, 2007) because of those problems.

Agile practices emerged as the way out for development coordination since they became mainstream in the industry and for distributed large-scale projects (TENDEDEZ; FERRARIO; WHITTLE, 2018; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009a). However, the need for agile practices in distributed large-scale projects must be accompanied by the awareness that tailoring the practices can not be simplified to using remote tools to mitigate the absence of physical contact. Rather, the teams' and organizations' specific context aspects must be considered. The teams' size, distribution, timezone differences, regulatory compliance, domain and technical complexity, company's distribution, history and discipline, and agile methodology in use may interfere with the agile tailoring journey of the teams (BROWN; AMBLER; ROYCE, 2013; CAMPANELLI; PARREIRAS, 2015). Due to it, the present study proposes the TARGET Framework as a collection of techniques and contexts description involving those aspects and being capable of easier the journey of agile teams through tailoring agile practices in various domains.

Practitioners and researchers must consider that different teams in different contexts will have unique needs and wills to tailor agile to their environment. Seen as a possible way to address the needs of the distributed teams using agile in large-scale settings, the organizations started to use agile and scaling agile frameworks to solve their issues regarding agile at scale.

The agile and scaling agile frameworks are capable of covering the gaps from agile methods for large-scale settings involving distributed teams or guiding companies in applying agile at scale through a set of best roles, practices, ceremonies, and workflow templates.

Undeniably, those frameworks have helped several distributed teams adopt agile practices in large-scale settings to achieve the project goals (ULUDAG et al., 2019; PAASIVAARA, 2017; RAZZAK et al., 2018; SALAMEH; BASS, 2019; SALAMEH; BASS, 2020; BROWN; AMBLER; ROYCE, 2013; HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b) in the middle of the barriers caused by the distributed environment and the large-scale dimension. Despite some success reports, most of those studies required a minimal tailoring approach in the frameworks, practices, roles, and concepts to accommodate the teams' needs. For example, Pandya *et al.* (PANDYA; MANI; PATTANAYAK, 2020) experience report describes a four-year SAFe transformation with specific tailoring techniques to accommodate the changes in the context of a consultancy firm. Meanwhile, Paasivaara presents a case study at an industry company to evaluate the adoption of SAFe on two business lines of the company (PAASIVAARA, 2017). In both contexts, teams from an industry organization and an IT consultancy firm required different tailoring approaches, i.e., the Release Train Engineer (RTE) in Paasivaara's cases (PAASIVAARA, 2017) involved an RTE working part-time and other full-time with different activities. At the same time, in the Pandya report, the RTE had closer responsibilities to what SAFe defines.

Each case study had specific context aspects related to distribution, culture, complexity, timezone, and business domain, leading them to different tailored approaches using the SAFe framework practices. Based on this, we reinforce the belief of Ambler and Lines that most of the several frameworks (HENRIK; ANDERS, 2012; Leffingwell, Dean, 2023; AMBLER; LINES, 2012; LARMAN; VODDE, 2016a; Sutherland, Jeff and Brown, Alex, 2021; Schwaber, Ken and Sutherland, Jeff, 2022) summarize best practices, events, and ceremonies while avoiding context understanding and the uniqueness of the organizations and teams. Without tailoring the set of practices defined by those frameworks, the teams can suffer from accessing a set of suggested techniques.

Based on such a scenario, researchers and practitioners looking forward to a better-structured manner to tailor agile in distributed teams on large-scale settings without depending entirely on the agile and scaling framework can access this study that presents the TARGET framework. Through the TARGET framework, we hope to provide a context-specific guide to help organizations according to their uniqueness during the agile tailoring process. Considering the context aspects of the teams and their business domain, scale dimension, and agile framework in use, the TARGET framework provides a collection of practices and different ways of applying it based on real industry cases from the literature.

In our study, the framework concept used to develop the TARGET comes from the descriptions and definitions of Petersen *et al.* (PETERSEN et al., 2008) and Wieringa *et al.* (WIERINGA et al., 2006). Frameworks are commonly used to structure the world investigated during a study (WIERINGA et al., 2006), gathering goals, objects, processes, and methods. Further, frameworks are also used to classify a set of identified practices from a set of empirical studies (PETERSEN et al., 2008). Based on it, we believe the TARGET framework aligns well with the framework concept from the literature.

The agile tailoring studies identified from the SLR can be considered quite mature. Most of them use empirical research methods within a high degree of rigor presented, 4.2. Such a fact reinforces the claims and findings of those studies, but some are based on experience reports with results not settled on empirical evidence. Moreover, the evaluation research type was the predominant research facet of the studies. Those results show a widespread number of studies investigating techniques in practice. However, most practices come from unstructured sources evaluated only by experts or surveys. A model such as the TARGET framework can elevate the maturity of those studies to assess the agile method tailoring techniques adopted by organizations in a more structured approach.

Our biggest motivation for the TARGET framework construction is to provide a handbook guide with similar purposes to the DA (LINES; AMBLER, 2019). But it specifically focuses on a research field that constantly requires more attention to address the challenges of agile tailoring in distributed large-scale projects. By covering 17 market sectors, three scale dimensions, five agile frameworks, and several context aspects that led teams to tailor an agile practice, the TARGET framework does not present only different techniques to address software development tasks. It offers several ways to apply the same agile practice according to the uniqueness and aspects of different teams, organizations, and studies. TARGET covers several possible paths which researchers and practitioners can consider to accommodate their needs.

The TARGET framework is based on the state-of-the-art of agile tailoring in distributed large-scale settings using agile frameworks. Through 74 studies from 17 market sectors and different scale dimensions, practitioners can consult the framework to get an overview of how different contexts applied some specific agile practices.

A manager can start the usage of the TARGET framework by organizations and teams, SM, PO, or even a regular team member that perceives the need for agile tailoring. Whether one of those roles wants to implement or consult an agile practice, the TARGET framework will serve as a handbook, i.e., if an SM from an industrial organization adapting the Scrum to a large and distributed context is in doubt on how to conduct a retrospective meeting. They can consult the TARGET framework structure for the general industry section and visualize three different studies from the same market sector, also in a large-scale dimension using Scrum, that had adapted it (HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b). By doing this, the SM can consult the agile practice details in the SLR results to evaluate which of the tailored approaches most adheres to its contexts and test the technique on his environment.

For example, some cases choose to conduct a common retrospective meeting for all the teams (PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b) in a common conference call. Still, the same study of Paasivaara *et al.* (PAASIVAARA; LASSENIUS, 2010) had reported over time that the SMs opted to conduct a separate retrospective meeting without the teams' members. On the other side, another study reported that the teams were supposed to execute a retrospective meeting for each of them at the end of the sprints and then report the results in the wiki (HOSSAIN; BANNERMAN; JEFFERY, 2011). But, over time, as the operation ran smoother, the teams perceived the need to conduct the retrospective after two iterations (HOSSAIN; BANNERMAN; JEFFERY, 2011). By accessing that information, the SM can judge which approach suits its context better and try it within the team. Whether it fits the organization's environment, minimal adjustments can be necessary to address specific needs not covered by the studies and contexts of our results. Moreover, a practitioner can also access the TARGET framework to better understand what aspects led those cases to constantly tailor the retrospective meeting over time.

The TARGET framework can also be used to evaluate the applicability of some agile tailored practices in other scale dimensions. For example, the same SM from a general industry can consult how several industrial companies had been tailoring the daily Scrum meetings in three different scale dimensions, from small-scale agile distributed teams to large-scale and very large-scale teams. By doing it, he can access the approaches closer to his context and other scale dimensions that can help him and his team. Industrial cases had applied several techniques for daily meetings beyond their scale dimensions but based on their needs and the teams' relationship, from dailies through teleconference tools to dailies twice a week, through chat messages, in consecutive schedules to allow managers participation, and even the absence of dailies events (BASS, 2014; HOSSAIN; BANNERMAN; JEFFERY, 2011; PAASIVAARA; LASSENIUS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2009b).

Moreover, the TARGET framework can also be used to evaluate whether a tailored approach from a team or area is beneficial regarding the literature results from the SLR. For example, whether a large-scale financial organization using the Spotify model within several distributed teams is looking forward to evaluating its approach regarding KPIs in a distributed large-scale setting. The managers can consult the TARGET framework to access other financial companies that have applied it. In this case, only the study by Bass and Salameh (SALAMEH; BASS, 2020) applied measurement indicators in agile distributed teams from a financial company using the Spotify model. However, the authors describe how the freedom to tailor KPIs damaged the monitoring of the teams' performance by the managers. Since the teams can choose their KPIs, comparing them with different data collected was impossible. Based on this result, practitioners from financial organizations using Spotify can realize that letting several teams free to choose their KPIs can harm the monitoring process, and based on the previous experience presented in the literature, a better approach may rely on defining a set of common KPIs for the teams, and then letting them defining their owns.

In a similar perspective, practitioners from the optical industry sector can access some tailored SAFe practices from Razzak *et al.* (RAZZAK et al., 2018) study covering a large-scale team that was not mature enough for some of the practices from the portfolio and teams levels. I.e., the PPPM team was not mature enough to work with strategic themes since they treated it informally without proper techniques for estimation and planning and were not using epic stories since the PPM team was used to acting as a regular team, focusing on projects and deliverables. Moreover, sprint contracts at the team level were not respected, with the manager introducing stories during the iteration and the teams rarely conducting retrospective meetings. Seen it, optical industry organizations can consult the TARGET framework to understand and learn how a low-mature team can impact the SAFe portfolio and program level implementation.

The TARGET framework benefits practitioners and researchers through beneficial ways of tailoring agile to handle the dynamic environment of different market sectors with distributed agile teams. Also, informing and describing how some tailored approaches among the agile frameworks studied can impact and harm the day-to-day activities. Due to it, researchers and practitioners can access the TARGET frameworks with the advantages of understanding good or not very good manners to tailor agile according to the needs and contexts of the studies. Similar to the strategy used by Ambler and Lines in the choose your way of working book (LINES; AMBLER, 2019), we aim to present a guided handbook framework that could be used to assess the reality of agile tailoring approaches gathering all types of manners and impacts of using agile techniques in distributed large-scale settings.

Regardless of the TARGET Framework's benefits, it is essential to highlight some limitations. The Spotify set of tailored approaches composed of 13 practices was extracted from only two studies, but the main limitation is that both studies are from the financial sector. Due to it, the coverage of our framework regarding agile distributed teams from large-scale settings tailoring the Spotify framework limits the financial industry within two case studies, reducing the practices' applicability to other sectors. Moreover, the DAD framework set of nine tailored practices is composed of a group of studies from three different market sectors: It service providers 4.8.1, software service providers 4.8.4, and financial 4.8.5. However, the case studies from those three market sectors are from very large-scale projects. Hence, the TARGET framework limits itself to tailored agile practices from distributed teams using DAD for very large-scale projects, lacking further investigation of DAD projects in small and large-scale dimensions. However, as we have seen in our case study, different scale dimensions and agile frameworks do not limit practitioners and researchers to evaluating tailored approaches from other sizes and trying them in their contexts.

An exciting fact identified while mapping the tailored agile practices is that daily meetings, usually common in any agile project despite the framework in use, were only seen in several studies using Scrum and in one study using DAD (BEECHAM et al., 2021). The point here is that Spotify (HENRIK; ANDERS, 2012), SAFe (Leffingwell, Dean, 2023), and LeSS (LARMAN; VODDE, 2016a) had a daily sync event similar to Scrum in its essence, but none of the studies from those scaling agile frameworks had described the use of it. This result does not limit the TARGET framework for those agile frameworks. Instead, organizations using them can look forward to the applications of Scrum and DAD to better understand how to implement the daily meeting.

The postmortem documentation process 4.3.7 seen in the literature and the BDD pipeline adaption seen in the case study 4.9.2.13 highlights that some practices are adapted as a replacement of the original practices due to the teams need. Since a meeting could not be held for the postmortem process, the team started documenting the postmortem activity. On the other side, since a design pipeline is unnecessary for a quality team, the QAs used their idea to make a pipeline of BDD.

From 95 tailored practices extracted from 74 studies that used five agile frameworks, only one practice was presented in every framework used by the studies, the Definition of Done

(DoD). The most exciting point of it, independently of the framework in use, at least one study had to tailor its checklist regarding the definition of done of its cards. The DoD of Spotify 4.3.6 comes to standardize the completeness process among the squads. For a similar purpose, the SAFe adoption of DoD seeks it in the teams of a program level 4.4.8. On DAD, the DoD concept focused on ensuring the quality of the deliverables. Still, it did not specify that a deliverable was ready to be consumed by the end users 4.5.3. Moreover, in LeSS, we could see the addition of the DoD definition, including the DoE and the DoR 4.6.9. The DoE describes the criteria for rough requirements to be broken into individual stories. In contrast, DoR describes the criteria the user stories must have to be considered ready for implementation. Finally, the DoD from Scrum gathers the highest number of studies for the practice, which DoD definition that goes from standard industries to highly regulated organizations 4.7.14.

Another popular tailored practice that was almost present in every framework was the Scrum of Scrums (SoS). Except for the DAD framework, we could not map a tailored technique for it. On Spotify, the term has changed to Squad-of-Squads meeting 4.3.11, but with similar purposes to the regular SoS meeting, to align all squads regarding issues about the behavior of new features released, progress, opportunities, and priorities. The questionable point is that Spotify squads seem independent, but the meeting was needed to accommodate customer and vendor relationship needs. On the other side, the other frameworks presented specific needs for tailoring the SoS meetings according to the context of the cases. The studies described tailoring approaches of the SoS meeting to handle better the customer-vendor relationships, suppliers coordination, and teams with significant timezone differences (See SAFe section 4.4.10, Less section 4.6.11, and Scrum 4.7.2).

It is also important to point out that the selected studies from our SLR began in 2007, six years later than the release of the agile manifesto. While other SLR studies regarding agile tailoring have shown studies since 2002 about the theme (CAMPANELLI; PARREIRAS, 2015). However, since our scope limits the agile tailoring to distributed teams on large-scale settings using agile frameworks, the year 2007 seems reasonable for the technological advances to be able to handle the needs of teams spread across the globe working remotely. Further, the following years of the 2010s elevated the number of studies on the research theme due to the emergence of the agile and scaling agile frameworks.

The TARGET framework, in contrast to the common agile frameworks in the market, can be used beyond the limits and borders of those structures. Whether there is no reference regarding adapting an agile practice in a specific market sector of interest, the researcher or practitioner can look forward to using these practices in a similar sector. I.e., sectors like automotive, optical, and oil and energy industries can benefit themselves by looking for practices not mapped to their sectors in the general industry sector of the TARGET framework and vice-versa. Since both sectors compose industrial and manufacturing needs, they can benefit from each other. It can also be seen on the internet and in telecommunication organizations, in which similar contexts can provide insights into tailored practices from different frameworks and scale dimensions. Such contrast of the TARGET framework goes beyond the borders of the market sectors. Organizations from different scales and using different agile frameworks could not limit themselves without consulting the practices of others companies.

Using the TARGET framework at the enterprise level can give enough autonomy for the organizations' teams and employees the power to choose, experiment, evaluate, test the results, and change their development process. It influences companies' culture and performance directly by letting them constantly try the best agile techniques that can accommodate their specific needs. The TARGET framework positions itself as the first structured framework artifact capable of providing enough inputs for distributed agile teams from large-scale settings requiring agile method tailoring in their daily routine.

5.2 THREATS TO VALIDITY

Several were the analysis and actions to mitigate the threats regarding the validity of the SLR, the TARGET framework, and the case study. As presented in this section, these perspective analyses and actions covered four types of validity threats (WOHLIN et al., 2012; AMPATZOGLOU et al., 2019). In this section, we discuss the following validity threats associated with the different activities of this study.

5.2.1 Construct Validity

Defines in what degree the operational measures that are studied represent what the researchers intended to look for and what is investigated according to the research question (WOHLIN et al., 2012). To reduce this, the SLR studies evaluations were conducted in a peer review approach through the data extraction process regarding the tailored agile practices. The advisors were involved during the agile tailored practices selection and description, and if any of them disagreed with the practice, they would discuss it to reach a consensus. Besides, during the case study, several data sources were used to reduce measurement bias through interviews with the team members, observation of the team's activities, and evaluation of the project documentation. Finally, we recognize that evaluating the TARGET framework in a single case involving one market sector from the 17 presented in the framework can cause mono-operation bias, but choosing the sector representing a third of the studies from the SLR may reduce such a threat.

5.2.2 External validity

It is related to what extent it is possible to generalize the findings and to what extent they have value to practitioners and researchers (WOHLIN et al., 2012). The SLR was conducted through a research protocol developed and validated by the author and the advisors. Also, we selected the most renowned databases for agile and distributed development research, such as ACM, IEEE Xplore, Springer, Scopus, and Wiley. The search string used in pre-defined bibliographic databases, which are references in agile development, ensures certain generalism of the findings since most studies in the area are published on those databases. Furthermore, another point that provides our external validity is that the research findings have interested practitioners and researchers since the middle of the 2000s (CHO, 2007). However, it is important to point out that the absence of the snowballing technique (STREETON; COOKE; CAMPBELL, 2004) in the SLR may increase the external validity threat. Finally, despite evaluating the TARGET framework through a single case study in the IT service provider sector, we reduce external validity by conducting the case in a real industry case involving employees among different roles, from developers to PO and quality analysts, working for an IT service provider company.

5.2.3 Conclusion Validity

It is concerned with to what extent the data and the analysis are consistent (WOHLIN et al., 2012). Also, it aims to ensure that the study assumptions and outcomes are not biased by the author's perspectives. Our study used fewer bibliographic databases, which may have made the research lose relative papers. However, to ensure conclusion validity, we selected the most renowned databases for agile research, such as ACM, IEEE, Springer, Scopus, and Wiley. Besides, the advisors revised the interview instrument and the project documentation to avoid searching for specific results regarding agile tailoring in the case study, similar to the findings in the literature. However, replicating the process of the SLR, the framework development and the case study can lead to different results due to the difference in individual decisions and evaluation of the linked practices among researchers.

5.2.4 Internal Validity

It is concerned with the effects of the treatments on the variables due to uncontrolled factors in the environment (WOHLIN et al., 2012). Following the research protocol, we aimed to mitigate this threat by analyzing the studies by the author and the advisors, analysis of the results, and redundancy. Besides, every study was chosen, and the involved researchers discussed the agile tailored practices extracted from them, and when one disagreed with a choice, all of them discussed it to reach a common agreement. Moreover, repeating the research protocol using the same search engines can ease the study reproduction. Finally, to reduce internal validity in the case study, the instruments used to extract data from the participants were reviewed by the advisors.

Despite the common validation threats that the study has. It is essential to point out some of the threats and mitigation actions taken based on the analysis of Ampatzoglu *et al.* (AMPATZOGLOU et al., 2019). Regarding the SLR, the study mapped bias, which can be caused due to conflicting inclusion/exclusion criteria, was mitigated by adding antonyms criteria. Besides, we did constant review cycles to validate and refine the agile tailored practices results to overcome the threat mentioned by Ampatzoglu *et al.* that may occur during the synthesis of the results. The same review cycles were constantly present while evaluating the agile practices used by the members from the case study.

6 CONCLUSION

The agile tailoring approach has proven popular in distributed agile teams of large-scale organizations. More and more organizations are adapting their teams' work to an agile flow that supports their dynamic and complex processes. Therefore, the one-size-fits-all agile approach, which is most common in agile and scaling agile frameworks, may not work since each organization, project, and team has its particularities and needs. Due to it, agile distributed teams and organizations must consider their unique needs and constraints when selecting and implementing agile practices to fit their context.

Based on it, this study contributes to the agile tailoring research theme by achieving its general and specific goals within an SLR summarization on five different bibliographic databases that started with 1520 studies in the first results of the research string. Then, it ended with 74 studies regarding the agile tailoring practices of distributed agile teams in large-scale contexts. The selected studies analyzed and extracted provided enough information about the state of the art of agile tailoring practices in DSD and large-scale environments. Five different agile and scaling agile frameworks were seen in those studies, and 95 tailored practices were mapped based on the literature. In addition, the present study has also shown how the 95 customized practices were tailored and implemented in several different contexts.

In our study, the contribution went beyond the SLR. Its results helped provide a systematic, methodological, and structured model, the TARGET framework. TARGET gathers the different market sectors, scale dimensions, agile and scaling agile frameworks, and the tailored practices used and extracted from the SLR studies. Putting it together, our study through TARGET provides a context-specific guide framework to help organizations during the agile tailoring process according to their uniqueness.

Moreover, to avoid staying in the field of theory, a case study was conducted in an agile distributed team from an IT service provider company in Brazil with more than one thousand employees to evaluate the TARGET Framework better. During the case, it was possible to perceive the similarities of the tailored practices from the case members within the TARGET framework part regarding the IT service provider sector with similar contexts. However, the particularities of the members and the project in the case have shown how different aspects of the context required tailored approaches not presented in the TARGET framework. This reinforces how different contexts will always require different tailored approaches.

As a contribution to academia and industry, this study provides a comprehensive overview of results regarding agile tailoring in distributed teams using agile and scaling agile frameworks from large-scale settings considering the uniqueness of those teams. Moreover, researchers and practitioners can access the adapted techniques presented here to understand better how different conditions can lead to different tailor techniques.

Despite the contributions of its study, it is important to explicit its limitations. We hope to evaluate the TARGET Framework in other industry cases, although it was impossible due to time and availability. Moreover, the SLR methodology can lead to the loss of many important studies since we did it through a single search string and did not perform snowballing. Finally, the TARGET framework was built based on market sectors, but some offered few tailored practices, which can harm the tailoring process of those companies.

Meanwhile, we believe the results and findings of this study will also enable researchers and practitioners that use agile and scaling agile frameworks to conduct further research and provide more dedicated solutions for the challenges in tailoring agile within the DSD context while using them.

6.1 FUTURE WORK

During the execution of this research, it is possible to point out some possibilities for future research that we intend to execute based on the results presented. During our work, we believe that we have accomplished a challenge related to tailoring agile practices in relation to the market sectors of the organization. By doing this, we hope other researchers and ourselves to explore the following possibility for future works are:

1. Evaluate the TARGET framework by executing other case studies to cover the other 16 market sectors presented on it and also in other real industry scenarios with other scale dimensions and agile frameworks;

2. Continue the SLR in the next years to expand the findings to cover other market sectors, practices, and particularities of organizations;

3. Execute a survey with distributed agile teams to verify whether the tailored practices of TARGET are being used.

4. Develop a more structured and visual representation of the TARGET framework to help practitioners and researchers visualize its practices and the contexts that applied it.
5. Examine the challenges faced by the projects as described in the literature, then evaluate how well the TARGET Framework can address those challenges.
6. Evaluate whether agile tailored techniques from one context can be well used in another domain.

We hope to accomplish a complete and mature overview of the TARGET Framework by conducting future works in the following years. Finally, to achieve a version of it that can serve as a guide for the industry and academia.

REFERENCES

ALQUDAH, M.; RAZALI, R. A review of scaling agile methods in large software development. *International Journal on Advanced Science, Engineering and Information Technology*, v. 6, p. 828–837, 2016.

AMBLER, S. W.; LINES, M. *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. 1st. ed. [S.l.]: IBM Press, 2012. ISBN 0132810131.

AMBLER, S. W.; LINES, M. Going beyond scrum: disciplined agile delivery. *Disciplined Agile Consortium. White Paper Series*, p. 1–16, 2013.

AMPATZOGLOU, A.; BIBI, S.; AVGERIOU, P.; VERBEEK, M.; CHATZIGEORGIOU, A. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Information and Software Technology*, v. 106, p. 201–230, 2019. ISSN 0950-5849. Available at: <https://www.sciencedirect.com/science/article/pii/S0950584918302106>.

AOYAMA, M. Agile software process and its experience. In: *Proceedings of the 20th International Conference on Software Engineering*. [S.l.: s.n.], 1998. p. 3–12.

BADAMPUDI, D.; FRICKER, S. A.; MORENO, A. M. Perspectives on productivity and delays in large-scale agile projects. In: BAUMEISTER, H.; WEBER, B. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 180–194. ISBN 978-3-642-38314-4.

BASS, J. M. Influences on agile practice tailoring in enterprise software development. In: *2012 Agile India*. [S.l.: s.n.], 2012. p. 1–9.

BASS, J. M. Agile method tailoring in distributed enterprises: Product owner teams. In: *2013 IEEE 8th International Conference on Global Software Engineering*. [S.l.: s.n.], 2013. p. 154–163.

BASS, J. M. Scrum master activities: Process tailoring in large enterprise projects. In: *2014 IEEE 9th International Conference on Global Software Engineering*. [S.l.: s.n.], 2014. p. 6–15.

BASS, J. M. How product owner teams scale agile methods to large distributed enterprises. *Empirical Softw. Engg.*, Kluwer Academic Publishers, USA, v. 20, n. 6, p. 1525–1557, dec 2015. ISSN 1382-3256. Available at: <https://doi.org/10.1007/s10664-014-9322-z>.

BASS, J. M. Artefacts and agile method tailoring in large-scale offshore software development programmes. *Information and Software Technology*, v. 75, p. 1–16, 2016. ISSN 0950-5849. Available at: <https://www.sciencedirect.com/science/article/pii/S0950584916300350>.

BASS, J. M. Large-scale offshore agile tailoring: Exploring product and service organisations. In: *Proceedings of the Scientific Workshop Proceedings of XP2016*. New York, NY, USA: Association for Computing Machinery, 2016. (XP '16 Workshops). ISBN 9781450341349. Available at: <https://doi.org/10.1145/2962695.2962703>.

BATRA, D.; VANDERMEER, D.; DUTTA, K. Extending agile principles to larger, dynamic software projects: A theoretical assessment. *J. Database Manage.*, IGI Global, USA, v. 22, n. 4, p. 73–92, oct 2011. ISSN 1063-8016. Available at: <https://doi.org/10.4018/jdm.2011100104>.

BECK, K.; BEEDLE, M.; BENNEKUM, A. van; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. *Manifesto for Agile Software Development*. 2001. Available at: <http://www.agilemanifesto.org/>.

BECK, K.; GAMMA, E. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000. (An Alan R. Apt Book Series). ISBN 9780201616415. Available at: <https://books.google.com.br/books?id=G8EL4H4vf7UC>.

BEECHAM, S.; CLEAR, T.; LAL, R.; NOLL, J. Do scaling agile frameworks address global software development risks? an empirical study. *Journal of Systems and Software*, v. 171, p. 110823, 2021. ISSN 0164-1212. Available at: <https://www.sciencedirect.com/science/article/pii/S0164121220302181>.

Betz, S.; Makio, J.; Stephan, R. Offshoring of software development - methods and tools for risk management. In: *International Conference on Global Software Engineering (ICGSE 2007)*. Munich, Germany: IEEE, 2007. p. 280–281.

BROWN, A. W.; AMBLER, S.; ROYCE, W. Agility at scale: Economic governance, measured improvement, and disciplined delivery. In: *Proceedings of the 2013 International Conference on Software Engineering*. [S.l.]: IEEE Press, 2013. (ICSE '13), p. 873–881. ISBN 9781467330763.

CAMARA, R.; ALVES, A.; MONTE, I.; MARINHO, M. Agile global software development: A systematic literature review. In: *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020. (SBES '20), p. 31–40. ISBN 9781450387538. Available at: <https://doi.org/10.1145/3422392.3422411>.

CAMARA, R.; MARINHO, M.; SAMPAIO, S.; HONóRIO, I.; MOURA, H. Outsourcing with distributed teams in large-scale environments. In: *Anais do XXV Congresso Ibero-Americano em Engenharia de Software*. Porto Alegre, RS, Brasil: SBC, 2022. p. 218–232. ISSN 0000-0000. Available at: <https://sol.sbc.org.br/index.php/cibse/article/view/20974>.

CAMARA, R.; MARINHO, M.; SAMPAIO, S.; HONóRIO, I.; MOURA, H. Outsourcing with distributed teams in large-scale environments. In: *Anais do XXV Congresso Ibero-Americano em Engenharia de Software*. Porto Alegre, RS, Brasil: SBC, 2022. p. 218–232. ISSN 0000-0000. Available at: <https://sol.sbc.org.br/index.php/cibse/article/view/20974>.

CAMARA, R.; MARINHO, M. L.; SAMPAIO, S.; CADETE, S. How do agile software startups deal with uncertainties by covid-19 pandemic? *International Journal of Software Engineering & Applications*, v. 11, p. 15–34, 07 2020.

CAMPANELLI, A. S.; PARREIRAS, F. S. Agile methods tailoring – a systematic literature review. *Journal of Systems and Software*, v. 110, p. 85–100, 2015. ISSN 0164-1212. Available at: <https://www.sciencedirect.com/science/article/pii/S0164121215001843>.

CARMEL, E.; AGARWAL, R. Tactical approaches for alleviating distance in global software development. *IEEE Software*, v. 18, n. 2, p. 22–29, 2001.

CESARE, S. de; IACOVELLI, N.; MERICO, A.; PATEL, C.; LYCETT, M. Tailoring software development methodologies in practice: A case study. *CIT*, v. 16, p. 157–168, 01 2008.

CHAPLIN. 1960. <https://www.charliechaplin.com/en/quotes>. [Online; accessed 15-March-2023].

CHO, J. Distributed scrum for large-scale and mission-critical projects. In: . [S.l.: s.n.], 2007. v. 1, p. 235.

CONBOY, K.; COYLE, S.; WANG, X.; PIKKARAINEN, M. People over process: key people challenges in agile development. 2011.

CONBOY, K.; FITZGERALD, B. Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion. *ACM Trans. Softw. Eng. Methodol.*, Association for Computing Machinery, New York, NY, USA, v. 20, n. 1, jul 2010. ISSN 1049-331X. Available at: <https://doi.org/10.1145/1767751.1767753>.

CRUZES, D. S.; DYBA, T. Recommended steps for thematic synthesis in software engineering. In: *2011 International Symposium on Empirical Software Engineering and Measurement*. [S.l.: s.n.], 2011. p. 275–284.

DANEVA, M.; van der Veen, E.; AMRIT, C.; GHAISAS, S.; SIKKEL, K.; KUMAR, R.; AJMERI, N.; RAMTEERTHKAR, U.; WIERINGA, R. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of Systems and Software*, v. 86, n. 5, p. 1333–1353, 2013. ISSN 0164-1212. Available at: <https://www.sciencedirect.com/science/article/pii/S0164121212003536>.

Denning, Steve. *What Does It Mean To Scale Agile?* 2021. <https://www.forbes.com/sites/stevedenning/2016/04/15/what-does-it-mean-to-scale-agile/?sh=53d099c778b9>. [Online; accessed 18-January-2021].

Digital.Ai, Inc. *16th Annual State of Agile Development Survey*. 2023. <https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf>. [Online; accessed 15-March-2023].

DIKERT, K.; PAASIVAARA, M.; LASSENIUS, C. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, v. 119, p. 87–108, 2016. ISSN 0164-1212. Available at: <https://www.sciencedirect.com/science/article/pii/S0164121216300826>.

DINGSØYR, T.; FÆGRI, T. E.; ITKONEN, J. What is large in large-scale? a taxonomy of scale for agile software development. In: *PROFES*. [S.l.: s.n.], 2014.

DORAIRAJ, S.; NOBLE, J.; ALLAN, G. Agile software development with distributed teams: Senior management support. In: *2013 IEEE 8th International Conference on Global Software Engineering*. [S.l.: s.n.], 2013. p. 197–205.

DORAIRAJ, S.; NOBLE, J.; MALIK, P. Understanding team dynamics in distributed agile software development. In: WOHLIN, C. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 47–61. ISBN 978-3-642-30350-0.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. *Information and software technology*, Elsevier, v. 50, n. 9-10, p. 833–859, 2008.

DYBA, T.; DINGSOYR, T.; HANSSEN, G. K. Applying systematic reviews to diverse study types: An experience report. In: *1st Int'l Conference on Empirical Software Engineering and Measurement (ESEM) 2007*. Madrid, Spain: IEEE, 2007. p. 225–234.

EBERT, C.; GALLARDO, G.; HERNANTES, J.; SERRANO, N. Devops. *IEEE Software*, v. 33, n. 3, p. 94–100, 2016.

EBERT, C.; PAASIVAARA, M. Scaling agile. *IEEE Software*, v. 34, n. 6, p. 98–103, 2017.

EDISON, H.; WANG, X.; CONBOY, K. Comparing methods for large-scale agile software development: A systematic literature review. *IEEE Transactions on Software Engineering*, v. 48, n. 8, p. 2709–2731, 2022.

EWUSI-MENSAH, K. Software development failures : anatomy of abandoned projects. In: . [S.l.: s.n.], 2003.

FITZGERALD, B.; RUSSO, N. L.; O'KANE, T. An empirical study of system development method tailoring in practice. In: *European Conference on Information Systems (ECIS)*. [s.n.], 2000. p. 187–194. Available at: <https://api.semanticscholar.org/CorpusID:7540031>.

FITZGERALD, B.; STOL, K.-J.; O'SULLIVAN, R.; O'BRIEN, D. Scaling agile methods to regulated environments: An industry case study. In: IEEE. *2013 35th International Conference on Software Engineering (ICSE)*. [S.l.], 2013. p. 863–872.

FITZGERALD, B.; STOL, K.-J.; O'SULLIVAN, R.; O'BRIEN, D. Scaling agile methods to regulated environments: An industry case study. In: *2013 35th International Conference on Software Engineering (ICSE)*. [S.l.: s.n.], 2013. p. 863–872.

GARBAJOSA, J.; YAGÜE, A.; GONZALEZ, E. Communication in agile global software development: An exploratory study. In: MEERSMAN, R.; PANETTO, H.; MISHRA, A.; VALENCIA-GARCÍA, R.; SOARES, A. L.; CIUCIU, I.; FERRI, F.; WEICHHART, G.; MOSER, T.; BEZZI, M.; CHAN, H. (Ed.). *On the Move to Meaningful Internet Systems: OTM 2014 Workshops*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 408–417. ISBN 978-3-662-45550-0.

GLASER, B. *Emergence Vs Forcing: Basics of Grounded Theory Analysis*. Sociology Press, 1992. (Emergence vs. forcing). ISBN 9781884156007. Available at: <https://books.google.com.br/books?id=1ZJiQgAACAAJ>.

GLASER, B.; STRAUSS, A. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine Transaction, 2009. (Observations (Chicago, Ill.)). ISBN 9780202363370. Available at: <https://books.google.com.br/books?id=rtiNK68Xt08C>.

GODOY, C. P.; SANTOS, L. M.; CRUZ, A. F.; ZERBINI, R. S.; SILVA, E. P.; PAHINS, C. A. L. Blueprint model: A new approach to scrum agile methodology. In: *Proceedings of the 14th International Conference on Global Software Engineering*. IEEE Press, 2019. (ICGSE '19), p. 85–89. Available at: <https://doi.org/10.1109/ICGSE.2019.00014>.

GUPTA, R. K.; JAIN, S.; SINGH, B. Challenges in scaling up a globally distributed legacy product: A case study of a matrix organization. In: *2018 IEEE/ACM 13th International Conference on Global Software Engineering (ICGSE)*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 77–81. ISBN 9781450357173. Available at: <https://doi.org/10.1145/3196369.3196389>.

GUPTA, R. K.; MANIKREDDY, P.; ARYA, K. C. Pragmatic scrum transformation: Challenges, practices & impacts during the journey a case study in a multi-location legacy software product development team. In: *Proceedings of the 10th Innovations in Software Engineering Conference*. New York, NY, USA: Association for Computing Machinery, 2017. (ISEC '17), p. 147–156. ISBN 9781450348560. Available at: <https://doi.org/10.1145/3021460.3021478>.

GUPTA, R. K.; VENKATACHALAPATHY, M.; JEBERLA, F. K. Challenges in adopting continuous delivery and devops in a globally distributed product team: A case study of a healthcare organization. In: *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. [S.l.: s.n.], 2019. p. 30–34.

HANSSEN, G. K.; ŠMITE, D.; MOE, N. B. Signs of agile trends in global software engineering research: A tertiary study. In: IEEE. *2011 IEEE Sixth International Conference on Global Software Engineering Workshop*. [S.l.], 2011. p. 17–23.

HENDERSON-SELLERS, B.; RALYTE, J. Situational method engineering: State-of-the-art review. *J. UCS*, v. 16, p. 424–478, 06 2010.

HENRIK, K.; ANDERS, I. *Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds*. 2012.

HILLEGERSBERG, J. van; LIGTENBERG, G.; AYDIN, M. N. Getting agile methods to work for cordys global software product development. In: KOTLARSKY, J.; WILLCOCKS, L. P.; OSHRI, I. (Ed.). *New Studies in Global IT and Business Service Outsourcing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 133–152. ISBN 978-3-642-24815-3.

HOBBS, B.; PETIT, Y. Agile methods on large projects in large organizations. *Project Management Journal*, v. 48, p. 3–19, 06 2017.

HODA, R.; KRUCHTEN, P.; NOBLE, J.; MARSHALL, S. Agility in context. *SIGPLAN Not.*, Association for Computing Machinery, New York, NY, USA, v. 45, n. 10, p. 74–88, oct 2010. ISSN 0362-1340. Available at: <https://doi.org/10.1145/1932682.1869467>.

HODA, R.; NOBLE, J. Becoming agile: A grounded theory of agile transitions in practice. In: *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 2017. (ICSE '17), p. 141–151. ISBN 9781538638682. Available at: <https://doi.org/10.1109/ICSE.2017.21>.

HOLE, S.; MOE, N. B. A case study of coordination in distributed agile software development. In: O'CONNOR, R. V.; BADDOO, N.; SMOLANDER, K.; MESSNARZ, R. (Ed.). *Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 189–200. ISBN 978-3-540-85936-9.

HOSSAIN, E.; BABAR, M. A.; VERNER, J. Towards a framework for using agile approaches in global software development. In: BOMARIUS, F.; OIVO, M.; JARING, P.; ABRAHAMSSON, P. (Ed.). *Product-Focused Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 126–140. ISBN 978-3-642-02152-7.

HOSSAIN, E.; BANNERMAN, P. L.; JEFFERY, R. Towards an understanding of tailoring scrum in global software development: A multi-case study. In: *Proceedings of the 2011 International Conference on Software and Systems Process*. New York, NY, USA: Association for Computing Machinery, 2011. (ICSSP '11), p. 110–119. ISBN 9781450307307. Available at: <https://doi.org/10.1145/1987875.1987894>.

HOSSAIN, S. S. Challenges and mitigation strategies in reusing requirements in large-scale distributed agile software development: A survey result. In: ARAI, K.; BHATIA, R.; KAPOOR, S. (Ed.). *Intelligent Computing*. Cham: Springer International Publishing, 2019. p. 920–935. ISBN 978-3-030-22868-2.

IVARSSON, M.; GORSCHEK, T. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering*, v. 16, p. 365–395, 06 2011.

Jalali, S.; Wohlin, C. Agile practices in global software engineering - a systematic map. In: *2010 5th IEEE International Conference on Global Software Engineering*. Princeton, NJ, USA: IEEE, 2010. p. 45–54.

JALALI, S.; WOHLIN, C. Global software engineering and agile practices: A systematic review. *Journal of Software: Evolution and Process*, v. 24, 10 2012.

JHA, M. M.; VILARDELL, R. M. F.; NARAYAN, J. Scaling agile scrum software development: Providing agility and quality to platform development by reducing time to market. In: *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. [S.l.: s.n.], 2016. p. 84–88.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing systematic literature reviews in software engineering*. [S.l.], 2007.

KOCH, C.; JØRGENSEN, C.; OLSEN, M.; TAMBO, T. We all know how, don't we? on the role of scrum in it-offshoring. In: BERGVALL-KÅREBORN, B.; NIELSEN, P. A. (Ed.). *Creating Value for All Through IT*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. p. 96–112. ISBN 978-3-662-43459-8.

KOMMEREN, R.; PARVIAINEN, P. Philips experiences in global distributed software development. *Empirical Softw. Engg.*, Kluwer Academic Publishers, USA, v. 12, n. 6, p. 647–660, dec 2007. ISSN 1382-3256. Available at: <https://doi.org/10.1007/s10664-007-9047-3>.

KORHONEN, K. Migrating defect management from waterfall to agile software development in a large-scale multi-site organization: A case study. In: ABRAHAMSSON, P.; MARCHESI, M.; MAURER, F. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 73–82. ISBN 978-3-642-01853-4.

KORKALA, M.; ABRAHAMSSON, P. Communication in distributed agile development: A case study. In: *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)*. [S.l.: s.n.], 2007. p. 203–210.

KORKALA, M.; PIKKARAINEN, M.; CONBOY, K. Distributed agile development: A case study of customer communication challenges. In: ABRAHAMSSON, P.; MARCHESI, M.; MAURER, F. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 161–167. ISBN 978-3-642-01853-4.

KRUCHTEN, P. Contextualizing agile software development. *Journal of software: Evolution and Process*, Wiley Online Library, v. 25, n. 4, p. 351–361, 2013.

KUMAR, K.; WELKE, R. J. Methodology engineeringr: A proposal for situation-specific methodology construction. In: _____. *Challenges and Strategies for Research in Systems Development*. USA: John Wiley &; Sons, Inc., 1992. p. 257–269. ISBN 0471931756.

KUSSMAUL, C. Onshore and offshore outsourcing with agility: Lessons learned. In: ______. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 91–106. ISBN 978-3-642-12442-6. Available at: <https://doi.org/10.1007/978-3-642-12442-6_6>.

LAL, R.; CLEAR, T. Enhancing product and service capability through scaling agility in a global software vendor environment. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 59–68. ISBN 9781450357173. Available at: <https://doi.org/10.1145/3196369.3196378>.

LARMAN, C.; VODDE, B. Scaling lean & agile development: Thinking and organizational tools for large-scale scrum. In: . [S.l.: s.n.], 2008.

LARMAN, C.; VODDE, B. *Large-scale Scrum: More with LeSS*. Addison-Wesley, 2016. (A Mike Cohn signature book). ISBN 9780321985712. Available at: <https://books.google.com.br/books?id=oZ_vngEACAAJ>.

LARMAN, C.; VODDE, B. *Large-scale scrum: More with LeSS*. [S.l.]: Addison-Wesley Professional, 2016.

LAUKKANEN, E.; LEHTINEN, T. O.; ITKONEN, J.; PAASIVAARA, M.; LASSENIUS, C. Bottom-up adoption of continuous delivery in a stage-gate managed software organization. In: *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery, 2016. (ESEM '16). ISBN 9781450344272. Available at: <https://doi.org/10.1145/2961111.2962608>.

LAUTERT, T.; NETO, A.; KOZIEVITCH, N. A survey on agile practices and challenges of a global software development team. In: ______. [S.l.: s.n.], 2019. p. 128–143. ISBN 978-3-030-36700-8.

LEE, J. C.; JUDGE, T. K.; MCCRICKARD, D. S. Evaluating extreme scenario-based design in a distributed agile team. In: *CHI '11 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2011. (CHI EA '11), p. 863–877. ISBN 9781450302685. Available at: <https://doi.org/10.1145/1979742.1979681>.

LEE, S.; YONG, H.-S. Distributed agile: Project management in a global environment. *Empirical Software Engineering*, v. 15, p. 204–217, 04 2010.

Leffingwell, Dean. *Scaled Agile Framework 6.0*. 2023. *https://scaledagileframework.com*. [Online; accessed 15-March-2023].

LEHTINEN, T. O.; VIRTANEN, R.; HEIKKILÄ, V. T.; ITKONEN, J. Why the development outcome does not meet the product owners' expectations? In: LASSENIUS, C.; DINGSØYR, T.; PAASIVAARA, M. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Cham: Springer International Publishing, 2015. p. 93–104. ISBN 978-3-319-18612-2.

LINDERS, B. *Don't copy the spotify model*. 2016. <https://www.infoq.com/news/2016/10/no-spotify-model/>. [Online; accessed 19-June-2022].

LINES, M.; AMBLER, S. *Choose Your WoW!: A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working (WoW)*. Independently Published, 2019. (Choose Your WoW Series). ISBN 9781790447848. Available at: <https://books.google.com.br/books?id=gTd4wAEACAAJ>.

LOUS, P.; TELL, P.; MICHELSEN, C. B.; DITTRICH, Y.; KUHRMANN, M.; EBDRUP, A. Virtual by design: How a work environment can support agile distributed software development. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 102–111. ISBN 9781450357173. Available at: <https://doi.org/10.1145/3196369.3196374>.

MAREK, K.; WIŃSKA, E.; DĄBROWSKI, W. The state of agile software development teams during the covid-19 pandemic. In: PRZYBYŁEK, A.; MILER, J.; POTH, A.; RIEL, A. (Ed.). *Lean and Agile Software Development*. Cham: Springer International Publishing, 2021. p. 24–39. ISBN 978-3-030-67084-9.

MARINHO, M.; NOLL, J.; BEECHAM, S. Uncertainty management for global software development teams. In: *11th International Conference on the Quality of Information and Communications Technology*. Coimbra, Portugal: IEEE, 2018. p. 238–246.

MARINHO, M.; NOLL, J.; RICHARDSON, I.; BEECHAM, S. Plan-driven approaches are alive and kicking in agile global software development. In: *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Porto de Galinhas, Brazil: IEEE, 2019. p. 1–11.

MARTINI, A.; BOSCH, J. A multiple case study of continuous architecting in large agile companies: Current gaps and the caffea framework. In: *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*. [S.l.: s.n.], 2016. p. 1–10.

MARTINI, A.; PARETO, L.; BOSCH, J. Communication factors for speed and reuse in large-scale agile software development. In: *Proceedings of the 17th International Software Product Line Conference*. New York, NY, USA: Association for Computing Machinery, 2013. (SPLC '13), p. 42–51. ISBN 9781450319683. Available at: <https://doi.org/10.1145/2491627.2491642>.

MARUPING, L. M. Implementing extreme programming in distributed software project teams: Strategies and challenges. In: _____. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 11–30. ISBN 978-3-642-12442-6. Available at: <https://doi.org/10.1007/978-3-642-12442-6_2>.

MATTHIESEN, S.; BJØRN, P. When distribution of tasks and skills are fundamentally problematic: A failure story from global software outsourcing. *Proc. ACM Hum.-Comput. Interact.*, Association for Computing Machinery, New York, NY, USA, v. 1, n. CSCW, dec 2017. Available at: <https://doi.org/10.1145/3139336>.

MERRIAM, S. *Qualitative Research: A Guide to Design and Implementation*. John Wiley & Sons, 2009. (Higher and adult education series). ISBN 9780470283547. Available at: <https://books.google.com.br/books?id=tvFICrgcuSIC>.

MOE, N. B.; ŠMITE, D.; ŠUNDEFINEDBLIS, A.; BöRJESSON, A.-L.; ANDRéASSON, P. Networking in a large-scale distributed agile project. In: *Proceedings of the 8th ACM/IEEE*

International Symposium on Empirical Software Engineering and Measurement. New York, NY, USA: Association for Computing Machinery, 2014. (ESEM '14). ISBN 9781450327749. Available at: <https://doi.org/10.1145/2652524.2652584>.

NOLL, J.; RAZZAK, A.; RICHARDSON, I.; BEECHAM, S. Agile practices for the global teaming model. In: *2016 IEEE 11th International Conference on Global Software Engineering Workshops (ICGSEW)*. [S.l.: s.n.], 2016. p. 13–18.

NOORDELOOS, R.; MANTELI, C.; VLIET, H. V. From rup to scrum in global software development: A case study. In: *2012 IEEE Seventh International Conference on Global Software Engineering*. [S.l.: s.n.], 2012. p. 31–40.

NYRUD, H.; STRAY, V. Inter-team coordination mechanisms in large-scale agile. In: *Proceedings of the XP2017 Scientific Workshops*. New York, NY, USA: Association for Computing Machinery, 2017. (XP '17). ISBN 9781450352642. Available at: <https://doi.org/10.1145/3120459.3120476>.

PAASIVAARA, M. Adopting safe to scale agile in a globally distributed organization. In: *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*. [S.l.: s.n.], 2017. p. 36–40.

PAASIVAARA, M.; DURASIEWICZ, S.; LASSENIUS, C. Distributed agile development: Using scrum in a large project. In: *2008 IEEE International Conference on Global Software Engineering*. [S.l.: s.n.], 2008. p. 87–95.

PAASIVAARA, M.; DURASIEWICZ, S.; LASSENIUS, C. Using scrum in distributed agile development: A multiple case study. In: *2009 Fourth IEEE International Conference on Global Software Engineering*. [S.l.: s.n.], 2009. p. 195–204.

PAASIVAARA, M.; DURASIEWICZ, S.; LASSENIUS, C. Using scrum in distributed agile development: A multiple case study. In: *2009 Fourth IEEE International Conference on Global Software Engineering*. [S.l.: s.n.], 2009. p. 195–204.

PAASIVAARA, M.; HEIKKILä, V. T.; LASSENIUS, C. Experiences in scaling the product owner role in large-scale globally distributed scrum. In: *2012 IEEE Seventh International Conference on Global Software Engineering*. [S.l.: s.n.], 2012. p. 174–178.

PAASIVAARA, M.; LASSENIUS, C. Using scrum practices in gsd projects. In: ______. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 259–278. Available at: <https://doi.org/10.1007/978-3-642-12442-6_17>.

PAASIVAARA, M.; LASSENIUS, C. Communities of practice in a large distributed agile software development organization-case ericsson. *Information and Software Technology*, v. 56, n. 12, p. 1556 – 1577, 2014. ISSN 0950-5849. Special issue: Human Factors in Software Development. Available at: <http://www.sciencedirect.com/science/article/pii/S0950584914001475>.

PAASIVAARA, M.; LASSENIUS, C. Scaling scrum in a large globally distributed organization: A case study. In: *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. [S.l.: s.n.], 2016. p. 74–83.

PAASIVAARA, M.; LASSENIUS, C.; HEIKKILä, V. T. Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work? In: *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. [S.l.: s.n.], 2012. p. 235–238.

PAASIVAARA, M.; VÄÄTTÄNEN, O.; HALLIKAINEN, M.; LASSENIUS, C. Supporting a large-scale lean and agile transformation by defining common values. In: DINGSØYR, T.; MOE, N. B.; TONELLI, R.; COUNSELL, S.; GENCEL, C.; PETERSEN, K. (Ed.). *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*. Cham: Springer International Publishing, 2014. p. 73–82. ISBN 978-3-319-14358-3.

PANDYA, A.; MANI, V. S.; PATTANAYAK, A. Expanding the responsibility of an offshore team and sustainably increasing business value using safe. In: *Proceedings of the 15th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020. (ICGSE '20), p. 1–5. ISBN 9781450370936. Available at: <https://doi.org/10.1145/3372787.3390441>.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. Swindon, GBR: BCS Learning & Development Ltd., 2008. (EASE'08), p. 68–77.

POPPENDIECK, M.; POPPENDIECK, T. Lean software development: An agile toolkit. *ACM*, Addison-Wesley, 2003.

POPPENDIECK, M.; POPPENDIECK, T. *Implementing lean software development: From concept to cash*. [S.l.]: Pearson Education, 2007.

RAHY, S.; BASS, J. Information flows at inter-team boundaries in agile information systems development. In: THEMISTOCLEOUS, M.; CUNHA, P. Rupino da (Ed.). *Information Systems*. Cham: Springer International Publishing, 2019. p. 489–502. ISBN 978-3-030-11395-7.

RAJPAL, M. Effective distributed pair programming. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE 18), p. 6–10. ISBN 9781450357173. Available at: <https://doi.org/10.1145/3196369.3196388>.

RALPH, P.; SHPORTUN, P. Scrum abandonment in distributed teams: A revelatory case. In: . [S.l.: s.n.], 2013. ISBN 9788995217016.

RAMESH, B.; CAO, L.; MOHAN, K.; XU, P. Can distributed software development be agile? *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 49, n. 10, p. 41–46, Oct. 2006. ISSN 0001-0782.

RAZZAK, M. A.; NOLL, J.; RICHARDSON, I.; CANNA, C. N.; BEECHAM, S. Transition from plan driven to safe: Periodic team self-assessment. In: *Product-Focused Software Process Improvement*. [S.l.: s.n.], 2017. p. 573–585. ISBN 978-3-319-69926-4.

RAZZAK, M. A.; RICHARDSON, I.; NOLL, J.; CANNA, C. N.; BEECHAM, S. Scaling agile across the global organization: An early stage industrial safe self-assessment. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA:

Association for Computing Machinery, 2018. (ICGSE '18), p. 121–130. ISBN 9781450357173. Available at: <https://doi.org/10.1145/3196369.3196373>.

RIZVI, B.; BAGHERI, E.; GASEVIC, D. A systematic review of distributed agile software engineering. *J. Softw. Evol. Process*, John Wiley & Sons, Inc., USA, v. 27, n. 10, p. 723–762, oct 2015. ISSN 2047-7473. Available at: <https://doi.org/10.1002/smr.1718>.

ROLLAND, K. H. Scaling across knowledge boundaries: A case study of a large-scale agile software development project. In: *Proceedings of the Scientific Workshop Proceedings of XP2016*. New York, NY, USA: Association for Computing Machinery, 2016. (XP '16 Workshops). ISBN 9781450341349. Available at: <https://doi.org/10.1145/2962695.2962700>.

ROLLAND, K. H.; FITZGERALD, B.; DINGSØYR, T.; STOL, K.-J. Problematizing agile in the large: Alternative assumptions for large-scale agile development. In: *ICIS*. [S.l.: s.n.], 2016.

ROLLAND, K. H.; MIKKELSEN, V.; NÆSS, A. Tailoring agile in the large: Experience and reflections from a large-scale agile software development project. In: SHARP, H.; HALL, T. (Ed.). *Agile Processes, in Software Engineering, and Extreme Programming*. Cham: Springer International Publishing, 2016. p. 244–251. ISBN 978-3-319-33515-5.

RUNESON, P.; HöST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, Kluwer Academic Publishers, USA, v. 14, n. 2, p. 131–164, apr 2009. ISSN 1382-3256. Available at: <https://doi.org/10.1007/s10664-008-9102-8>.

SABLIS, A.; SMITE, D.; MOE, N. Team-external coordination in large-scale software development projects. *J. Softw. Evol. Process*, John Wiley & Sons, Inc., USA, v. 33, n. 3, mar 2021. ISSN 2047-7473. Available at: <https://doi.org/10.1002/smr.2297>.

SALAMEH, A.; BASS, J. Spotify tailoring for b2b product development. In: *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. [S.l.: s.n.], 2019. p. 61–65.

SALAMEH, A.; BASS, J. M. Heterogeneous tailoring approach using the spotify model. In: *Proceedings of the Evaluation and Assessment in Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2020. (EASE '20), p. 293–298. ISBN 9781450377317. Available at: <https://doi.org/10.1145/3383219.3383251>.

SCHEERER, A.; SCHIMMER, T.; KUDE, T. Coordination in large-scale agile software development: A multiteam systems perspective. *47th Hawaii International Conference on System Sciences (HICSS)*, 01 2014.

SCHWABER, K.; SUTHERLAND, J. The scrum guide: The definitive guide to scrum: The rules of the game. In: . [S.l.: s.n.], 2020.

Schwaber, Ken and Sutherland, Jeff. *The Scrum Guide*. 2022. *https://www.scrum.org/resources/scrum-guide*. [Online; accessed 06-August-2022].

SEKITOLEKO, N.; EVBOTA, F.; KNAUSS, E.; SANDBERG, A.; CHAUDRON, M.; OLSSON, H. H. Technical dependency challenges in large-scale agile software development. In: CANTONE, G.; MARCHESI, M. (Ed.). *Agile Processes in Software Engineering and*

Extreme Programming. Cham: Springer International Publishing, 2014. p. 46–61. ISBN 978-3-319-06862-6.

Sriram, R.; Mathew, S. K. Global software development using agile methodologies: A review of literature. In: *2012 IEEE International Conference on Management of Innovation Technology (ICMIT)*. Sanur Bali, Indonesia: IEEE, 2012. p. 389–393.

STREETON, R.; COOKE, M.; CAMPBELL, J. Researching the researchers: Using a snowballing technique. *Nurse researcher*, v. 12, p. 35–46, 02 2004.

Sutherland, Jeff and Brown, Alex. *Scrum@Scale*. 2021. <https://www.scrumatscale.com/scrum-at-scale-guide/>. [Online; accessed 16-July-2022].

TENDEDEZ, H.; FERRARIO, M. A. M.; WHITTLE, J. Software development and cscw: Standardization and flexibility in large-scale agile development. *Proc. ACM Hum. Comput. Interact.*, Association for Computing Machinery, New York, NY, USA, v. 2, n. CSCW, nov 2018. Available at: <https://doi.org/10.1145/3274440>.

ULUDAG, O.; KLEEHAUS, M.; DREYMANN, N.; KABELIN, C.; MATTHES, F. Investigating the adoption and application of large-scale scrum at a german automobile manufacturer. In: *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. [S.l.: s.n.], 2019. p. 22–29.

USMAN, M.; BRITTO, R.; DAMM, L.-O.; BöRSTLER, J. Effort estimation in large-scale software development: An industrial case study. *Information and Software Technology*, v. 99, p. 21–40, 2018. ISSN 0950-5849. Available at: <https://www.sciencedirect.com/science/article/pii/S0950584918300326>.

VÄLIMÄKI, A.; KÄÄRIÄINEN, J. Patterns for distributed scrum — a case study. In: MERTINS, K.; RUGGABER, R.; POPPLEWELL, K.; XU, X. (Ed.). *Enterprise Interoperability III*. London: Springer London, 2008. p. 85–97. ISBN 978-1-84800-221-0.

VALLON., R.; BAYRHAMMER., K.; STROBL., S.; BERNHART., M.; GRECHENIG., T. Identifying critical areas for improvement in agile multi-site co-development. In: INSTICC. *Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE,*. [S.l.]: SciTePress, 2013. p. 165–172. ISBN 978-989-8565-62-4. ISSN 2184-4895.

VALLON, R.; DRäGER, C.; ZAPLETAL, A.; GRECHENIG, T. Adapting to changes in a project's dna: A descriptive case study on the effects of transforming agile single-site to distributed software development. In: *2014 Agile Conference*. [S.l.: s.n.], 2014. p. 52–60.

VALLON, R.; ESTÁCIO, B.; PRIKLADNICKI, R.; GRECHENIG, T. Systematic literature review on agile practices in global software development. *Information and Software Technology*, v. 96, 12 2017.

VALLON, R.; STROBL, S.; BERNHART, M.; GRECHENIG, T. Inter-organizational co-development with scrum: Experiences and lessons learned from a distributed corporate development environment. In: BAUMEISTER, H.; WEBER, B. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 150–164.

WAHYUDIN, D.; HEINDL, M.; ECKHARD, B.; SCHATTEN, A.; BIFFL, S. In-time role-specific notification as formal means to balance agile practices in global software development settings. In: MEYER, B.; NAWROCKI, J. R.; WALTER, B. (Ed.). *Balancing Agility and Formalism in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 208–222. ISBN 978-3-540-85279-7.

WENGER, E.; MCDERMOTT, R.; SNYDER, W. *Cultivating Communities of Practice: A Guide to Managing Knowledge*. Harvard Business School Press, 2002. (NetLibrary Inc). ISBN 9781578513307. Available at: <https://books.google.com.br/books?id=m1xZuNq9RygC>.

WIERINGA, R.; MAIDEN, N.; MEAD, N.; ROLLAND, C. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.*, v. 11, p. 102–107, 03 2006.

WILDT, D.; PRIKLADNICKI, R. Transitioning from distributed and traditional to distributed and agile: An experience report. In: ______. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 31–46. ISBN 978-3-642-12442-6. Available at: <https://doi.org/10.1007/978-3-642-12442-6_3>.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M.; REGNELL, B.; WESSLÉN, A. Experimentation in software engineering. In: ______. [S.l.]: Springer, 2012. p. 123–151. ISBN 978-3-642-29043-5.

YIN, R. *Case Study Research: Design and Methods*. SAGE Publications, 2003. (Applied Social Research Methods). ISBN 9780761925521. Available at: <https://books.google.com.br/books?id=BWea_9ZGQMwC>.

ZAHEDI, M.; SHAHIN, M.; Ali Babar, M. A systematic review of knowledge sharing challenges and practices in global software development. *International Journal of Information Management*, v. 36, n. 6, Part A, p. 995–1019, 2016. ISSN 0268-4012. Available at: <https://www.sciencedirect.com/science/article/pii/S026840121630384X>.

ZAMBONI, A. B.; THOMMAZO, A. D.; HERNANDES, E. C. M.; FABBRI, S. C. P. F. Start uma ferramenta computacional de apoio à revisão sistemática. 2010.

APPENDIX A – SYTEMATIC LITERATURE REVIEW SHEET

The full details of the data analysis process and the results of the categorization process of the studies can be accessed through . The Google sheet concentrates on the studies information, the quality assessment grades for each criterion (DYBA; DINGSOYR; HANSSEN, 2007), the studies categorization among research and contribution facets (PETERSEN et al., 2008; WIERINGA et al., 2006), its type, and also the study method and data collection methods used. Moreover, the Google sheet also describes essential information regarding the studies, such as the number of people involved, the organization's size, scaling agile framework used, countries involved, company domain, type of distribution, and scaling dimension (DINGSØYR; FÆGRI; ITKONEN, 2014). Finally, the rigor and relevance evaluation followed by the extracted quotes are also presented in the Google sheet.