Agile tailoring in distributed large-scale environments using agile frameworks: A Systematic Literature Review

Rafael Camara
Federal University of Pernambuco, Informatics Center (CIn),
Recife, Brazil, 50740-560,
rcf8@cin.ufpe.br

and

Marcelo Marinho
Federal Rural University of Pernambuco, Computing Department (DC),
Recife, Brazil, 52171-900,
marcelo.marinho@ufrpe.br

and

Hermano Moura
Federal University of Pernambuco, Informatics Center (CIn),
Recife, Brazil, 50740-560,
hermano@cin.ufpe.br

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. To evaluate how large-scale agile distributed teams adapt their practices to meet their specific contexts, this study launches a Systematic Literature Review (SLR). The SLR presents adaptations of agile methodologies in distributed software development teams operating in large-scale environments. With the growing popularity of agile methodologies, there is an increasing need to adapt them to suit the specific needs of distributed teams operating in large-scale contexts. The review identified 96 adapted practices from five agile frameworks (Scrum, Scaled Agile Framework (SAFe), Large Scale Scrum (LeSS), the Spotify model, and Disciplined Agile Delivery (DAD)) used in various case studies between 2007 and 2021. Scrum was the most commonly adapted framework with 32 customized practices, followed by SAFe (25), LeSS (17), the Spotify model (13), and DAD (9). The review provides insights into how these practices have been tailored to meet the needs of distributed teams in large-scale contexts. The findings can guide organizations in adapting agile practices to their specific contexts.

Keywords: Agile, Distributed Software Development, Large-scale, Tailoring agile, Scaling agile frameworks, Agile frameworks, Systematic Literature Review.

1 Introduction

Even before the COVID-19 pandemic, companies pursued Distributed Software Development (DSD) to reduce production costs and expand their talent pool. This has been documented in various studies [1, 2] and is a common practice in the software industry ([3]). However, following the pandemic, remote work became necessary for many organizations, prompting them to seek ways to effectively manage the development of their software solutions across geographically dispersed teams [4].

Organizations are adopting agile methodologies to solve the challenges posed by distributed software development since DSD has become the mainstream process for software development [4]. According to the 16th Annual State of Agile Report and the survey by VersionOne, 80% of the respondents say their organizations are working with agile teams distributed geographically [5].

Agile practices for DSD shouldn't be limited to their purest form. Large-scale projects with distributed agile teams require a tailored and scalable approach [6]. Scaling agile frameworks have been developed to address this challenge by providing guidance for companies to apply agile methodologies in distributed and large-scale contexts. However, most of the agile frameworks [7–12] choose a one-size-fits-all approach to lead the agile adoption in large-scale scenarios, which may not work since each large-scale project, organization, and the team will have its particularities and needs [13, 14]. For handling it better, organizations are considering agile method tailoring for better management of the development process [6].

The discussion regarding 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 answer it, we conducted a Systematic Literature Review (SLR) with 74 studies from 2001 to 2021 on how distributed teams tailor agile practices using popular frameworks like Spotify, SAFe, DAD and DA, LeSS, and Scrum. The references were limited from 2001 onwards since the Agile Manifesto was published this year [15]. We identified 96 agile tailored practices and described how organizations tailored them.

The remainder of this study is organized as follows: In section 2, we introduce the background. Section 3 describes the proposed research method. Section 4 present the SLR results. Then, section 5 has the purpose of discussing the SLR findings. Finally, in section 6, we state some concluding remarks.

2 Background

2.1 Agile Distributed Software Development (ADSD)

Agile Distributed Software Development (ADSD) refers to applying agile practices, methodologies, and techniques on distributed projects [16]. Even though agile practices were initially developed to help co-located teams, numerous studies demonstrate that agile methods can help mitigate DSD problems [13, 17, 18]. Therefore, agile methods appear to be a viable option for implementing DSD as it is increasingly used in software development globally.

Numerous organizations utilize DSD broadly to tap into worldwide talent, decrease costs, attain economic benefits, expedite delivery, and facilitate 24-hour software development cycles because of varying time zones [19]. Nonetheless, though the adoption of distributed development has grown among companies, having team members in various locations can lead to communication, coordination, and control issues in the development process [20], which depends on the teams to customize their methods.

To tackle DSD challenges, daily meetings, planning, and pair programming practices are tailored to accommodate a distributed environment [20]. For instance, teams in different locations and time zones may also need to adjust face-to-face meetings and pair programming activities for remote contexts.

2.2 Large-scale Agile projects

The precise definition of "large-scale agile" is a topic of ongoing discussion. Dikert et al. [21] point out that "large-scale" can vary depending on the context and individual defining it. However, the number of teams involved is also a key factor in determining project scale. For this reason, This study utilizes the taxonomy presented by Dingsoyr *et al.* [22], which categorizes the scale of agile software development projects into three levels based on the number of teams involved. It suggests that small-scale projects typically have one team, large-scale projects can have 2 to 9 teams, and very large-scale projects may involve ten or more teams.

Various challenges exist when implementing agile methodologies in large projects. For instance, as the team grows, coordinating and synchronizing efforts becomes more complex [23]. Furthermore, it requires cultural and organizational changes from all stakeholders [24–26]. However, scaling frameworks can aid in achieving commitment during the adoption process [23, 27, 28].

2.3 Scaling Agile

Scaling agile applies agile methodologies to large and complex environments involving multiple teams, departments, and stakeholders [3]. 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. These frameworks provide guidelines and

best practices for coordinating the work of multiple teams, improving communication and collaboration, and ensuring alignment with business goals. The most recent State of Agile Report survey [5] indicates that the SAFe [8] remains the most popular scaling framework, with adaptations of Scrum [12] for large-scale settings. Although less prevalent, other scaling frameworks, such as DAD [9], LeSS [10], and the Spotify model [7], are also being used.

The precise definition of "large-scale agile" is a topic of ongoing discussion. Dikert et al. [21] point out that "large-scale" can vary depending on the context and individual defining it. However, the number of teams involved is also a key factor in determining project scale. For this reason, this study utilizes the taxonomy presented by Dingsoyr *et al.* [22], which categorizes the scale of agile software development projects into three levels based on the number of teams involved. It suggests that small-scale projects typically have one team, large-scale projects can have 2 to 9 teams, and very large-scale projects involve ten or more teams.

2.4 Agile Tailoring

The one-size-fits-all approach most present in scaling agile frameworks could lead organizations to lack flexibility, have inadequate communication, and limited skills to adapt to changes [13]. 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 [6].

Tailoring agile practices, principles, and values to fit the specific needs of a software development project, team, or organization is known as agile tailoring. This customized approach helps to increase the project's value, reduce risks and uncertainties associated with its context, and improve team performance and adaptability. Two classifications for method tailoring have been defined in the literature by Conboy and Fitzgerald [29, 30]: contingency factors and method engineering.

The contingency factors approach entails carefully choosing suitable principles, values, and practices from different methods that align with the specific needs of the organization and team [30]. Method engineering, on the other hand, entails developing a new method tailored to existing method fragments to the specific context rather than simply adopting standard practices from various methodologies [29].

2.5 Related Works

Campanelli and Parreiras conducted an SLR [6] on the tailoring of agile methods and its research field. The review aimed to understand the technical aspects and the research community's perspective and identify trends and gaps in the existing research. The review also highlighted several criteria for tailoring agile methods, but it does not address distributed agile teams in large-scale environments. Thus, further investigation is necessary in these domains.

Ambler and Lines, founders of DAD and DA, 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 agile practices. The emphasis on choosing your Way of Working (WoW) [14] is relevant to our agile tailoring research work since we also consider that the teams' and organizations' context and specificities drive the agile tailoring practices selection process. The book helped us during the extraction of the SLR results by providing clearance on how organizations can use different techniques and assess their effectiveness.

Bass interviewed 46 practitioners from 8 international companies to evaluate Scrum Master (SM) and Product Owner (PO) roles adaptations in large-scale contexts [31,32]. The studies provide detailed descriptions of how these roles are tailored in different sectors but don't cover how agile practices are tailored. We aim to expand on these findings by providing a broader overview of tailored practices in large-scale settings beyond just agile roles.

A study by Alqudah and Razali [33] compared six agile frameworks, analyzing their roles and practices and identifying similarities and differences. The study focused solely on the pure use of the frameworks and did not consider case studies regarding tailoring agile in the literature. 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 are tailored according to the agile team's needs.

3 Research Method

A Systematic Literature Review following the guidelines of Kitchenham and Stuart [34] 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.

Three researchers conducted the SLR. During the SLR's phases, the authors developed the research protocol, while the first author executed the search string in the bibliography databases. The search results were exported as BibTeX files, then organized into the StArt software [35], an open-source support tool for SLR research, and then all the articles were evaluated until the full read phase.

3.1 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 "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/3OZYLAX`) or in the table 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.

Table 1: 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")

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 [15]. 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 framework were selected.

3.1.1 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; (IC3) the study is available via the university library services; (IC4) keywords of the research string appear on abstract or author keywords; (IC5) studies directly related to agile tailoring in large-scale distributed settings.

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 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 DSD, scaling agile, tailoring agile methods, and large-scale software development.

We ensured no duplicates or replications before including any study in our research. A total of 90 duplicated studies and two replications were removed. After this, we moved on to the selection phase, where we found 247 studies in phase 1 based on abstract and title reading. We read the introduction and conclusion of each study, and 91 of them were selected for a full reading. In phase 2, we removed 17 studies that were not related to our subject matter. Finally, we were left with 74 studies and evaluated each one regarding agile tailoring practices used by distributed teams (See table 2). Any disagreements were resolved through discussion and reaching a consensus.

The first author reviewed study titles and abstracts, discussed the dataset with the other authors, and included studies approved by two or all researchers. Disagreements were resolved through discussion and reaching a consensus. Out of 247 studies in phase 1, the first author selected those meeting the criteria and

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

Table 2: Studies by engines.

extracted quotes from the final set of 74 studies that all researchers evaluated. 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.

3.2 Study Quality

Our study's quality assessment criteria are based on principles and good practices established for driving empirical research in software engineering [36]. 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?

3.3 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 [37]. 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. Every aspect is evaluated and given a score that determines a paper's rigor level. The scale consists of three values: 0 for "weak," 0.5 for "medium," and 1 for "high.". On the other hand, differently from rigor, industrial relevance is evaluated in four aspects that only accept a binary score, 1 for present and 0 for not. The aspects are considered in terms of the subjects, context, scale of applications, and research method. For example, case studies, surveys, and action research provide more relevant evidence than laboratory experiments or conceptual analyses. Use a scale of up to 4 to rate the level of relevance.

3.4 Data Extraction

Data extraction refers to recording all relevant information from the studies required to answer the RQ [38]. To synthesize the data and ease the management, we conducted some recommended steps by [39].

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 studies 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. [40] and Wieringa et al. [41].

Further, 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 [42].

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

[1] www.maxqda.com

distributed teams in large-scale environments.

4 Results

4.1 Overview of the studies

Our SLR results are from 74 studies [1–3, 13, 17–20, 23–28, 31, 32, 43–100] that helped us extract 96 tailored agile practices from DSD teams from 2007 to 2021. The 96 agile tailored practices were described following the structure: I) Name - Title of the practice; II) Goal - The aim 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. Out of the 96 practices identified across five popular frameworks, Spotify had 13 [7], SAFe had 25 [8], DAD had 9 [9], LeSS had 17 [10], and Scrum had 32 [12].

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 (10 studies). Finally, only three studies opted for fully quantitative approaches (See figure 1).

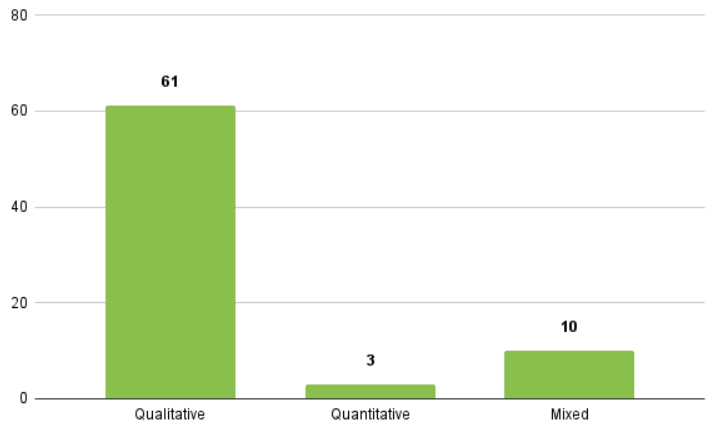

Figure 1: Research approaches from the studies.

The 74 studies were also evaluated according to their research type facets from Wieringa *et al.* [41]. The research type facets are derived from Wieringa *et al.* [41], and it aims to classify the papers regarding the research types. In the early 2000s, no studies on agile tailoring in large-scale distributed teams were found, despite agile being born at the start of the century. However, from 2007 to 2021, more studies emerged. 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 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 (See figure 2).

The distribution of contribution type facets of the reviewed studies derived from Petersen *et al.* [40] and is presented in Figure 3. 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.

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, 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.

Regarding the evaluation of the studies, each one was assessed on their aspects regarding rigor and industrial relevance [37]. As shown in figure 5, 20 out of 74 selected studies were classified with the highest

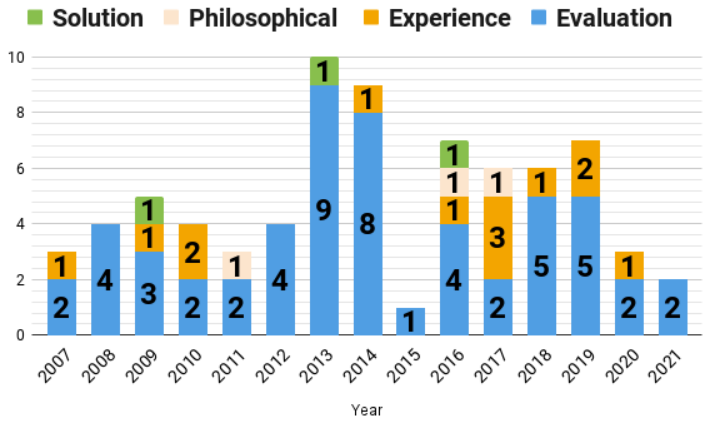

Figure 2: Research type facets over time.

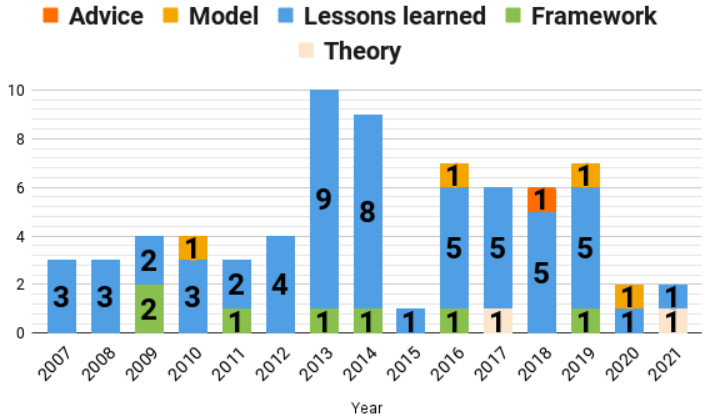

Figure 3: Contribution type facets over time.

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.

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 to have 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 6. 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.

The subsequent subsections aim to present the agile, tailored practices that have been identified in each scaling agile framework. Each title of the agile practice is accompanied by a number that indicates the frequency of studies that have employed and adopted the agile practice.

4.2 Spotify Tailoring Practices

The 13 Spotify agile tailoring practices originated from only two studies. Both of them are from the financial sector [27, 55]. Salameh and Bass [27] conducted an embedded case study to evaluate the Spotify Model's

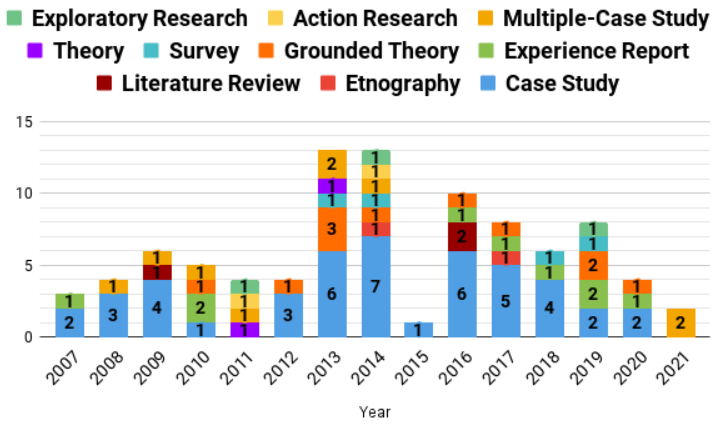

Figure 4: Research methods over time.

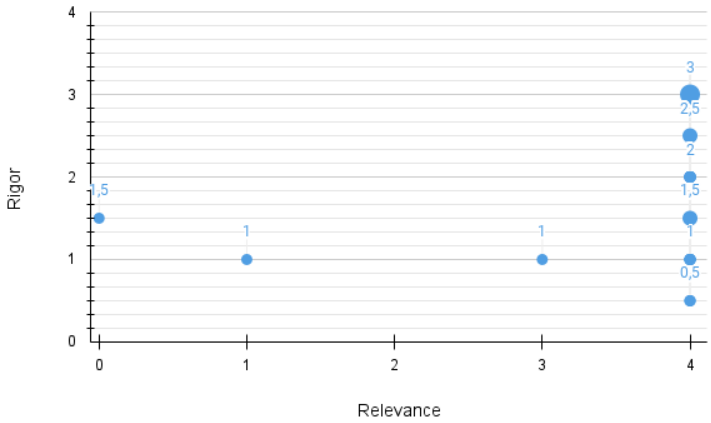

Figure 5: Rigor and Relevance of the studies.

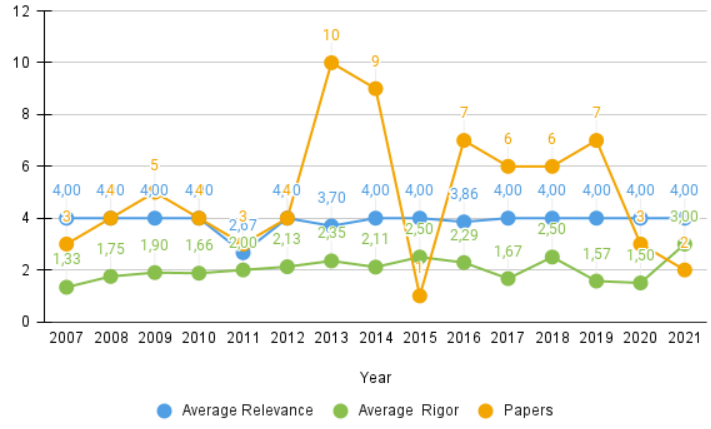

Figure 6: Rigor and Relevance over time.

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 [55].

4.2.1 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 [55]. **Who**: team members, PO, SM. **How**: Bass and Salameh presented two studies [27, 55] on a fintech organization that had squads with different missions, going from maintenance, product-line, and innovation to Proof of Concept (PoC) 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 [27,55]. However, squads responsible for developing new and complex features faced high uncertainty. Due to this, they combined Lean Startup and Kanban processes to handle the estimation and development [55]. Finally, squads working with mini-projects and PoCs were more uncertain than any other team. Due to this, 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 [55]. According to the authors, those squads considered estimation techniques a waste of activity for innovation teams. Therefore, sacrificing predictability was seen as more important for the sake of innovation and customer value [55]. **Context**: The study conducted by Salameh and Bass [55] was taken in a fintech organization that used Spotify in 6 different teams with a lot of heterogeneity since they chose different tailored approaches.

4.2.2 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 [27,55].**Who**: managers, team members.**How**: in one of the studies conducted by Bass and Salameh [55], 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 [55]. Furthermore, the support squad also represented the second level of contact with the customer, helping them with issues investigation and service configuration [27]. **Context**: both studies with support and maintenance squads were held in financial organizations [27, 55], which require a certain level of support for the customers and stability of the released versions since the tolerance for bugs is low.

4.2.3 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 [55]. **Who**: chapters, team members, managers, Product Owner (PO), and Scrum Master (SM). **How**: as presented by Salameh and Bass [27, 55], 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 [27, 55]. **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 [27, 55].

4.2.4 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 [27]. **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 [27]. The development of a common vision easily creates long- and short-term goals for the squads [55]. POs' and KAMs' should constantly communicate the project vision through regular meetings for the teams to achieve the solution strategy [27]. Despite the customer's intention, the vendor must be aware, guaranteeing that the product development does not deviate from their vision [55]. **Context**: due to the outsourcing nature presented in the studies of Salameh and Bass [27, 55], the need to report a clear vision of the solution became an important point to ensure the squads into the right direction.

4.2.5 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 [27]. 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 [27]. 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 [27]. However, when an incident occurred, the squad could roll back the changes and stabilize the environment [27]. **Context**: The study conducted by Salameh and Bass [27] 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 [27] and to avoid big issues in the whole user chain.

4.2.6 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 [27]. 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 [27] was handled in a financial company that used Spotify to develop a B2B solution for a large-scale mission-critical project.

4.2.7 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 [2]. In one of the studies, the postmortem meeting was tailored to be a postmortem documentation process [27]. **Who**: managers, team members. **How**: Due to the fail-friendly culture introduced by the Spotify framework [101], 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 [27]. The postmortem documentation was also filled with customer feedback that was used to improve the product and the development process [27]. **Context**: due to the business domain of the study [27], 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.

4.2.8 Measurement Indicators (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 [55]. **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 [55]. 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 [55]. 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 [55] evaluates the heterogeneity of agile tailoring in six different squads. The use of specific indicators in each squad has shown the damage caused in the monitoring process of the squads.

4.2.9 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 [55]. However, who should make these decisions? **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

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

architectural decisions among the squads [55]. According to the authors, the absence of such a process impacts the quality of the development of the solution. However, its presence may impact the squads' autonomy [55]. 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 [55]. **Context**: in the study presented by Bass and Salameh [55], the architect's role was required since the chapter leaders were not handling the architectural decisions. Further, the squads were not reaching a consensus.

4.2.10 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 [55]. **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" [7]. However, since the study published by Salameh and Bass [55] 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 [55]. **Context**: in 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 [55]. 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.

4.2.11 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 [27]. **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 [7]. However, Spotify squads should not usually hold such meetings since squads are quite independent and don't require this level of synchronization [7]. 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 [27]. 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 [27]. 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.

4.2.12 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 [27]. **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 [27]. **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 this, those features must always be usable.

4.2.13 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 [27]. 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 [102]. In this case, vendor transparency and respect fostered the relationship with the client.

4.3 SAFe Tailored Practices

The SAFe framework has the second highest number of practices in our study among seven papers. In total, 25 agile tailored practices were computed from studies of different market sectors. The market sectors are

the financial sector, healthcare companies, IT service providers, optical industry, and telecommunication organizations.

4.3.1 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 [8]. **Who**: every team, Release Train Engineer (RTE). **How**: in the case study presented by Razzak *et al.* [3], PI planning practices were not implemented effectively, and the project manager supported PI planning outputs without involving other team members. Paasivaara's case study [23] demonstrated efficient PI planning with presentations, team-specific planning, SoS meetings, retrospectives, and real-time communication [23]. Gupta *et al.* [65] tailored the PI planning event to a two-day workshop, where the teams refine the current backlog increments and discuss the backlog risks for the next two version increments. **Context**: Razzak et al. [3] used some elements of the PI planning process but missed important points like stakeholder participation. In the Paasivaara case study [23], teams were more committed, and both business lines were involved in PI planning, achieving good results. Gupta et al. [65] reported success in a healthcare project using PI planning, releasing all planned versions on time with high customer satisfaction.

4.3.2 External Coaches and Consultants (3)

Name: external coaches and consultants. **Goal**: SAFe framework suggests hiring a consultant for its adoption due to its specific events, roles, and functions [23]. **Who**: coaches and consultants. **How**: In Paasivaara's study [23], a consultant supported the adoption of SAFe through training, workshops, and coaching. The team appreciated the consultant's help in coaching the managers and practicing to improve. In another case from the same study, the team faced more problems because they started SAFe adoption without training. Further, Pandya *et al.* [28] reported that a consultant helped Scrum teams transition to SAFe by assessing current practices and fostering team dynamics. Lautert *et al.* [80] surveyed a large company that uses SAFe, and they asked about Agile and SAFe training and provided courses to those without training. **Context**: in the Paasivaara case study [23], the consultant played an essential role in the Paasivaara case study, whereas in the Pandya *et al.* [28] study, the consultant was indispensable during the organization's transition from Scrum to SAFe.

4.3.3 Content readiness (2)

Name: content readiness. **Goal**: in SAFe, content readiness is important to ensure a clear vision and context for every person in the PI Planning [8]. **Who**: team members responsible for the backlog writing. **How**: to achieve a good level of content readiness before a PI Planning, the teams must sufficiently prepare the product and architectural backlog [28]. In another study [65], the project manager was responsible for chasing content readiness at the beginning of the project version. **Context**: in both studies of Pandya *et al.* [28] and Gupta *et al.* [65], 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.

4.3.4 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 [3, 32, 89]. **Who**: staff members. **How**: in a case study presented by Razzak *et al.* [3] and Beecham *et al.* [89] 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 [3, 89]. In another study, Bass and Beecham [32] 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 [32]. **Context**: in the case study of Razzak *et al.* [3], 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 [32] interviewed practitioners from 8 different companies to map how the PO was scaling

agile in those large distributed agile projects. Due to the high number of people involved, staff members were necessary to keep things working.

4.3.5 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 are necessary to their environment [3]. **Who**: the company. **How**: in the case study published by Razzak *et al.* [3], 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 [3]. 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.* [3] 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.

4.3.6 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. **Who**: team members including quality manager, regulatory expert, and operation/back office team. **How**: according to the experience report published by Gupta *et al.* [65], the teams were conducting project increment workshops which are not well defined in SAFe [8]. As presented in the study, the activity was held for two days, rotating the locations among the teams' sites: India, USA, and Germany. Independent of the chosen location, quality, ops, and development, members would travel to 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 to help the teams in refining their process, and metrics and enabling a short release cycle with quality [65]. 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 [65]. **Context**: Gupta *et al.* [65] 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.

4.3.7 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.* [28], 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.* [28] consisted of an experience report regarding four years of transformation from a Scrum-based organization to the SAFe framework [8]. Those practices were required to achieve the business demands during the transformation better.

4.3.8 Definition of Done (DoD) (1)

Name: 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.* [28] 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 [8], the need to tailor some practices from the Scrum team level brings the attention of the program level presented in SAFe.

4.3.9 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.* [28], 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 [28].

4.3.10 Scrum of Scrums (SoS) (1)

Name: Scrum of Scrums (SoS). **Goal**: according to SAFe [8], 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 [23] at Comptel presented how a business line applied the SoS meetings. First, the SoS meetings were part of the 2-day PI planning event. At first, the teams discussed the plans for the architecture and the business vision. Then, one or two SoS meetings to check the status and coordinate the planning were held [23]. The RTE was responsible for coordinating and arranging regular SoS meetings. **Context**: Paasivaara, in this study, presents a case study at Comptel [23]. 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.

4.3.11 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 [8]. **Who**: quality analysts. **How**: in a multiple case study presented by Beecham *et al.* [89], a very large-scale project used automated tests to shorten its release cycle [89]. 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.* [89] 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.

4.3.12 Feature team (1)

Name: Feature Team. **Goal**: according to [8], 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.* [89], 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 this, 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.* [89], 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 could drive the solution to a balanced number of fixes and new features during the interactions [89]. Further, in this case, the company moved part-time product owners to full-time personnel who could focus solely on the PO role to reduce the impact of emergingrequirements.

4.3.13 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 [8]. However, a single product backlog comprises features, capabilities, and enablers in the experience report by Pandya *et al.* [28]. **Who**: team members, product owner. **How**: to achieve predictability through a high throughput of a product backlog that concentrates on 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 [28]. By doing it, the teams achieved a yearly release schedule, which helped acquire high predictability. **Context**: Pandya *et al.* [28] presented an experience report on a software development team in Idia. The report covers four years of transformation from a Scrum-based organization to the SAFe framework.

4.3.14 Measurement Indicators (KPIs) (1)

Name: measurement indicators. **Goal**: keeping track of the solution requires measurement indicators [8] 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.* [28], Key Performance Indicators (KPI) 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.* [28] 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.

4.3.15 Keep stakeholders close (1)

Name: keep stakeholders close. **Goal**: keeping track of key stakeholders' expectations is vital to the solution's success [28]. 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 [28]. 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 [28]. Each key stakeholder was interviewed and mapped to a four-quadrant structure considering their attitude and expectations [28]. **Context**: in this study [28], the stakeholders played an important role in the solution roadmap. New interviewers must deal with those stakeholders' changed expectations depending on the project phase. Due to this, keeping track of them was a matter of driving the solution to success.

4.3.16 Instructor-led training (1)

Name: instructor-led training. **Goal**: training team members in technical skills, process, and tools operation during the development process [28]. **Who**: mentor. **How**: in the study presented by Pandya *et al.* [28], 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 [28] through the reduction of available effort. **Context**: in this experience report [28], 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.

4.3.17 Strategic Themes (1)

Name: strategic themes. **Goal**: strategic themes play an important role in the portfolio decision-making process [8]. It provides a business context that helps connect the portfolio to the enterprise's strategy [8]. **Who**: Project Portfolio Management (PPM), directors, board. **How**: according to SAFe, Objectives and Key Results (OKRs) are used during the development of strategic themes or even sentences that can influence everyone in the solution development [8]. However, in the Razzak *et al.* [3] 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 [3]. **Context**: in the study [3], 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.

4.3.18 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 a Minimum Viable Product (MVP), also describes a significant initiative of development that covers different values streams, program increments, and financial approval before implementation [8]. **Who**: PPM team. **How**: the CTO from the company studied at Razzak *et al.* [3] 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 in the team's environment [3]. **Context**: in the study [3], the PPM maturity level was low, which caused the lack of important practices such as epics in the portfolio management area.

4.3.19 Sprints (1)

Name: sprints. **Goal**: sprints are cycle time iterations in which teams compromise to deliver specific software increments for the solution of an ART [8]. **Who**: teams. **How**: in the same case study presented by Razzak

et al. [3], 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 this, the team put the tickets without a proper estimation, which hindered the team from achieving the iteration goal [3]. 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.* [3] 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.

4.3.20 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 [8]. **Who**: team members. **How**: according to SAFe [8], 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.* [3]. One of the developers reported that only one retrospective during the last two years was done after a release and not a simple iteration [3]. **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.* [3] did not discuss further implications regarding retrospectives.

4.3.21 User stories (1)

Name: user stories. **Goal**: similar to SCRUM [12], stories in SAFe are short descriptions of a feature that needs to be implemented [8]. It also may describe the functionality in the user's language. **Who**: product owner. **How**: the case study presented by Razzak *et al.* [3] 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 [3]. 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 [3]. 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 [3] 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.

4.3.22 ART for Business Lines (1)

Name: ART for business lines. **Goal**: The Agile Release Train ART is a combination of agile teams working with stakeholders to develop and deliver incrementally and operating one or more solutions in a value stream [8]. **Who**: teams, stakeholders. **How**: Passiavaara, in her case study [23], 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 [23]. 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 [23] was evaluating the adoption of SAFe in two different business lines of Comptel, a huge telecommunication company with employees worldwide. 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.

4.3.23 SAFe adoption (1)

Name: SAFe adoption. **Goal**: Some issues may arise when adopting any framework, including SAFe. However, the teams must focus on solving those issues fast [23]. **Who**: the whole organization. **How**: Paasivaara, in her study [23] has evaluated the adoption of SAFe in two different cases of the same company. The first case adopted SAFe without paying much attention to the issues during the adoption. Some people reported unhappy concerns during the retrospective, but the managers did nothing to fix it [23]. In the other case, the problems regarding the adoption were solved immediately to keep the teams satisfied [23]. Unlike the first scenario, the second case used retrospective meetings to develop action plans with responsible persons assigned to those actions to improve the SAFe implementation. Due to it, the people involved were happy even when facing problems [23]. **Context**: despite the good approach from the second case to handle issues from the SAFe adoption, it was easier to do things this way since the first case at the same company

suffered problems with it [23]. Further, assigning responsibilities and monitoring the implementation of the action plans made the improvements visible.

4.3.24 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 [44]. 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.* [23, 44] studies, the authors had presented the function of change agents. In the first case of [23], the change agent was the RTE who 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 [23] and in the other study [44] 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 [23, 44]. **Context**: as we have seen, in the Paasivaara *et al.* [23, 44] studies, the second case took more advantage by having all the information regarding the adoption of the first case. Due to this, the second case better addressed the issues while involving change agents to achieve success.

4.3.25 Release Train Engineer (RTE) (1)

Name: Release Train Engineer (RTE). **Goal**: according to SAFe, the RTE is responsible for facilitating the major events of the ART [8]. They are also responsible for assisting the process and coaching the team to deliver value [8]. **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 [8]. However, in the first case from Paasivaara's study [23], 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 [23]. 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 [23]. Moreover, he was responsible for leading the PI Planning, the SoS meetings, and taking care of the improvement items continuously [23]. 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 [23]. **Context**: the second case from Paasiavaara's study [23] 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.

4.4 DAD Tailored Practices

The DAD framework studies seen in the literature could provide only 9 different agile tailored practices. Despite the low popularity, it was possible to extract interesting findings from agile practices of other frameworks in studies using DAD. Within a total of only three studies, the DAD studies covered the market sectors of software service providers, IT service providers, and the financial sector.

4.4.1 Risk Mitigation (1)

Name: risk mitigation. **Goal**: DAD uses a risk-value driven lifecycle approach [9]. Risk-related features are high-priority, not high-value. This ensures that delivery-related risks are addressed early on. **Who**: the teams and the organizations. **How**: Beecham *et al.* [89] studied two organizations that used DAD [9] and SAFe [8] frameworks to address GSD risks. The original DAD practices covered GSD risks, but only half of the catalog risks were observed in the DAD organization [89]. **Context**: Beecham *et al.* [89] found that a case company successfully developed asset management software with ten teams spread across three countries using DAD for five years. The organization's mature DAD application enabled them to address every GSD risk through a risk- and value-driven approach.

4.4.2 Spikes (1)

Name: spike. **Goal**: Spikes validate technical approaches before committing to a design choice related to architecture [9]. **Who**: developers, architectures, and development team members. **How**: In the study conducted by Beecham *et al.* [89], the company reported the use of spikes to mitigate some risks while using DAD. Initially, the teams were developing automated tests for some components they were unfamiliar with.

As a result, spikes were used to address this challenge. **Context**: the company's use of DAD represents a mature approach to its practices [89]. Beyond architectural prototyping, they used Spikes to develop stories and achieve business goals.

4.4.3 Definition of Done (DoD) (1)

Name: Definition of Done (DoD). **Goal**: DAD framework doesn't define DoD concept but discusses it from the lean community's perspective. According to the lean community, a work item is considered done only when it's delivered to the user and they are satisfied. [9]. **Who**: the team. **How**: In a case study [58], a company transitioned to DAD working on a high-quality solution aimed to have zero bug interaction. They used DoD practice, including governance practice, continuous delivery, and pair programming [58]. However, the study did not focus on end-user consumption.**Context**: Lal and Clear studied a global software vendor for 15 years, observing 10 teams transition from RUP to Scaled Agile with DAD. They evaluated the changes in practices, roles, and responsibilities due to agile dynamics [58].

4.4.4 Daily Tactical Huddle (1)

Name: daily tactical huddle. **Goal**: the dailies from DAD do not differ much from regular dailies of Scrum [12]. 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 [9]. 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 instead of the individual [9]. **Who**: the team. **How**: Beecham *et al.* [89] has shown that the case using DAD implements 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 tasks were required since the organization worked in the program and project divisions. 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 [89]. **Context**: the case company using DAD on Beecham *et al.* [89] works with several projects that together deliver the program. Due to this, the program activities were held globally with leadership who had 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, which discussed day-to-day work items.

4.4.5 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 [9]. However, DAD does not prescribe user stories specifically. It recommends a usage-driven approach with a requirement artifact focused on usage, like user stories, usage scenarios, or even use cases [9]. **Who**: the team, PO, and analysts. **How**: Beecham *et al.* [89] 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 underlying problem of conflict requirements and unclarified requirements [89]. In his concern, the user stories are important to keep the workflow, although the most interesting insights happen in the day-to-day adjustments [89]. 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.* [89] study develops asset management software for enterprise demands. The focus on requirement stability is understandable, although the customers' needs are fluid. Due to this, it is important to consider spending some effort on good requirements elicitation to avoid further issues.

4.4.6 Integration and Unit Testing (1)

Name: integration and unit testing. **Goal**: DAD frameworks encourage the use of integration and unit testing [9]. 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 [9]. 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 [9]. Conversely, integration tests on large and distributed teams may be complex, and DAD suggests having an independent test team for this [9]. The integration tests focus on verifying the potential defects that have fallen through regular unit testing [9]. It also aims to execute preproduction testing to ensure quality before a release [9]. **Who**: development team, testers, and independent test team. **How**: Brown *et al.* [60] 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 [60]. 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 [60]. **Context**: Brown *et al.* [60] 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, and 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.

4.4.7 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 [9]. 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 [9]. **Who**: cross-functional team members. **How**: Lal and Clear [58] 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 [58]. 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.

4.4.8 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 [9]. The portfolio planning identifies potential new projects and evaluates the dependencies between the ongoing ones, but more details are beyond the framework's scope [9]. It also happens to program/product plans, which look forward to two or three releases and business goals that it will accomplish [9]. Neither of those plans is the focus of DAD [9]. **Who**: the organization, leadership. **How**: in the Lal and Clear [58] 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 [9]. Finally, the responsibilities shared between the Sales and engineering sectors encouraged a better link between the necessary development skills and collaborative management [58]. **Context**: the study of Lal and Clear [58] 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 this, the teams became closer, and the organization drove the solution to a common vision and roadmap.

4.4.9 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 the Lal and Clear [58] study, the company has hired the proper founder of the DAD framework to coach the teams and actively facilitate the scaled agile transition. Due to this, 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 [58]. **Context**: the study of Lal and Clear [58] 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.

4.5 LeSS Tailored Practices

The LeSS framework has the third highest number of practices in our study, with 17 practices. Only two studies were tailoring LeSS generated the 17 agile tailored practices representing the automotive and

telecommunication industry.

4.5.1 Community of Practice (CoP) (2)

Name: CoP. **Goal**: 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 [103]. **Who**: team members. **How**: in the Paasivaaraand Lassenius [1] 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 [1] and the lack of self-organization. However, in Uludag *et al.* [43] multiple-case study, the four products evaluated organized the CoPs for collaboration and information exchange involving technical and business subjects. Architects and stakeholders set up a CoP to discuss design guidelines affecting feature teams [43]. Further, teams created CoPs for POs, SMs, and testers to facilitate communication and knowledge-sharing [43]. **Context**: two studies, from Uludag et al. [43] and Paasivaara and Lassenius [1], describe the development of a complex product by multiple teams located across the world. The first study cites the formation of CoPs as instrumental in building the product, as team members self-organized around relevant topics [43]. In contrast, the second study notes that the traditional mindset of the teams impeded knowledge sharing [1].

4.5.2 Requirement Area (RA) (2)

Name: RA. **Goal**: LeSS requires a requirement area structured around customer requirements. Each area has an Area Product Owner who supervises 4-10 feature teams working on the same backlog, divided by product boundaries [1, 10, 43]. **Who**: the organization, teams, and APOs. **How**In a case study, Uludag *et al.* [43] used RAs similar to the LeSS Huge framework and had one APO for each RA, while in another case study at Nokia, they required different requirement areas, but having one APO for each area was not feasible [1]. This caused features to cross several areas, which broke the framework [1].**Context**: Nokia's lack of experience and the complexity of the project led to several teams working on the same features, creating chaos. Due to a tight schedule, Architect APOs were responsible for designing tasks for more than eight teams [1].

4.5.3 Area Product Backlog (2)

Name: area product backlog (APB). **Goal**: APBs are product backlog artifacts from requirement areas managed by the APO. Each PB is associated with one Area Backlog [10]. **Who**: APOs. **How**: Uludag et al. [43] found that the APO maintains an area product backlog, similar to LeSS. One product had six specific backlogs and a common backlog, while three products focused on one backlog. Nokia's case study [1] had 20 teams working on a common backlog in an Excel sheet. **Context**: Uludag et al.'s study found one product requiring six specific backlogs due to its complexity [43]. In contrast, Nokia's case study showed that a common backlog initially worked for their teams across four countries. However, as the project grew, they needed RAs and APB, and the teams reorganized accordingly [1].

4.5.4 LeSS Huge (1)

Name: LeSS Huge. **Goal**: LeSS provides two different large-scale Scrum frameworks [10]. 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 [10]. **Who**: the organization. **How**: Uludag *et al.* [43] 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 [43]. The teams' justification was that the subordinate products do not have an overarching product character that would require a LeSS Huge implementation [43]. From another product perspective, using 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 [43]. To solve this, the organization developed an APO with six product owners responsible for subareas as suggested by LeSS Huge1. **Context**: Uludag *et al.* [43] 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.

4.5.5 Single-Specialist Teams (1)

Name: single-specialist teams. **Goal**: in LeSS, the scrum teams should be cross-competence teams with multi-disciplinary skills [10]. It also explicitly states that single-specialist teams may not exist since the team should have all the necessary competencies [10]. **Who**: the organization. **How**: in Paasivaara and

Lassenius [1] 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 [1]. Further, the company established verification teams in Germany and Greece to deal with network checks and performance testing [1]. Moreover, the authors reported the presence of a software architecture team with nine specialists responsible for supporting the teams in architecture planning [1]. Finally, a maintenance team composed of two managers and 1-2 members from each team is built. This team aims to handle trouble reports from external customers and internal platforms [1]. **Context**: despite the recommendations of LeSS Framework [10]. The complexity of the case study of Paasivaara and Lassenius [1] required the organization to build multi-disciplinary scrum teams and work with single-specialist teams. The study was conducted at Nokia, and it involved people from India, Germany, Finland, and Greece who were developing many products for the telecommunication industry. The complexity of some parts of the solution required single-specialist teams [1].

4.5.6 Inspect and Adapt (1)

Name: inspect and adapt. **Goal**: inspect and adapt are particularly a practice of SAFe [8], 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.* [43], two of four products that were adopting LeSS conducted regular inspection 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 [43]. 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.* [43] study were using LeSS with SAFe [8] and LeSS with DevOps [104]. 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.

4.5.7 Design And Requirement Workshops (1)

Name: design and requirement workshop. **Goal**: design and requirement workshops are helpful to clarify story aspects for the teams [10]. During the activity, the team members must discuss the inconsistencies the system should support and describe the story through workflow visualizations and abstract descriptions [10]. **Who**: APO's and team members. **How**: during the case at Nokia from the study of Paassivaara and Lassenius [1], 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 requirements in detail and how the System and Software Architects had decided to implement them at a high level [1]. 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 [1]. However, in reality, the user stories were so big that they required those workshops always and took three sprints to be implemented [1]. 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 [1]. The teams also said the APO could hear their opinions regarding the design. However, in a rush to release new features, some were misplanned in the workshops, which caused issues while implementing, like new requirements or forgetting of planning details [1].

4.5.8 Retrospective Meeting (1)

Name: retrospective meeting. **Goal**: in LeSS [10], the retrospective meetings should 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 [10]. LeSS also has overall retrospectives with team representatives, scrum masters, POs, and managers to discuss general actions and issues [1]. **Who**: team members, POs, SMs, POs, managers. **How**: the Paasivaara and Lassenius [1] study at Nokia has shown several applications of retrospective meetings. In the beginning, each team had its retrospective meeting and needed to describe three issues that were big enough for them [1]. 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 [1]. Due to this, the organization tried implementing a different common sprint retrospective with an internal coach to avoid past mistakes. In this

new way, the team members must brainstorm the most important impediments, choose the most important one, search for root causes, brainstorm solutions, choose one, and draft an implementation of it [1]. 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 [1]. 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.

4.5.9 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 [10]. 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 [43]. **Who**: teams. **How**: in Uludag *et al.* [43] 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 [43]. 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 [43]. Moreover, the case study has shown other definition techniques, such as the Definition of Entry (DoE) and the Definition of Entry (DoR). The DoE describes rough requirements into individual stories, while the DoR describes the last step before implementation that represents user stories ready for implementation [43]. **Context**: Uludag *et al.* [43] study presents different products with different levels 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 [43].

4.5.10 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 [10]. **Who**: team members, POs, APOs, stakeholders, and other interested people. **How**: the study of Paasivaara and Lassenius [1] has described the evolution of a simple demo presentation activity. In the beginning, a common demo meeting of two hours replaced a common sprint review [1]. 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 [1]. 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 [1]. **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 [1]. Since the number of teams evolved, the common meeting became impossible in two hours, and the individual one seemed more efficient.

4.5.11 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 [10]. 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 [10]. **Who**: team's representatives, scrum masters. **How**: Paasivaara and Lassenius [1] 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 [1]. 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 [1]. 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 [1]. **Context**: the case study at Nokia [1] had 20 teams with 170 members spread across India, Finland, Germany, and Greece [1]. In the beginning, a common SoS meeting seemed enough, although by the time the teams grew, some alternatives may have taken 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 [1].

4.5.12 Teams Representatives (1)

Name: teams representatives. **Goal**: many meetings from LeSS are usually held in common with the different teams [10]. Further, not all members must attend these meetings, and representatives are elected to participate [10]. **Who**: team's representatives. **How**: in the case study of Paasivaara and Lassenius at Nokia, the [1] the rotation of team 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 [1]. 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 [1]. Further, some representatives started to miss the common meetings, which indicates that fixed team representatives can be seen as a burden for some members [1].

4.5.13 Sprint Planning (1)

Name: sprint planning. **Goal**: LeSS framework defines sprint planning as two-part event [10]. 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 [10]. In the second phase, each team conducts its sprint planning, in which they create their plan to get the items done during the sprint [10]. **Who**: PO, SM, and all teams.**How**: Paasivaara and Lassenius's study at Nokia [1] has shown some adaptions to the sprint planning event. Similar to LeSS [10], 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 [1]. In this meeting, the PO would discuss the market situation and present and assign the user stories for the teams. After that, the sprint planning continues in each team space for detailed planning [1]. Finally, in the evening, each team emailed the committed items to the PO and program manager, who updated the backlog [1]. **Context**: the common sprint planning was conducted at Nokia due to many teams involved [1]. 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.

4.5.14 Releases (1)

Name: releases. **Goal**: since the LeSS framework is based on Scrum [10], 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 [1] 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 [1]. After that, the company established two major software releases yearly and six maintenance releases with no new functionality or bug fixing. Meanwhile, the project became more mature and was released once per month for the main customers [1]. **Context**: in this study at Nokia [1], 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 [1], 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.

4.5.15 Area Product Owner (APO) (1)

Name: APO. **Goal**: according to the LeSS framework [10], 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 [10]. 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 [10]. **Who**: POs, APOs. **How**: in the Nokia case study by Paasivaara and Lassenius [1], 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 [1]. The System architect belongs to the R&D organization sector and is responsible for the technical demands and architectural plans [1]. Meanwhile, the solution architect belongs to the product management sector, which can have a business and technical background [1]. 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 [1]. The APO would have a couple of development teams, and each team would be responsible for developing a feature [1]. However, while establishing requirement areas, the issues blocked it 4.5.2. **Context**: the use of Less Huge in the study of Paasivaara and Lassenius at Nokia [1] involving 20 teams required the presence of APOs as

suggested by the framework [10]. 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 [1].

4.5.16 System and Solution Architects (1)

Name: system and solution architects. **Goal**: originally, LeSS [10] does not define the system and solution architects' role, although one article combined both to represent APOs. In LeSS, an APO focuses on customer-centric requirement areas as a PO for the teams of those areas [10]. One study **Who**: APO. **How**: in the Paasivaara and Lassenius [1] 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 its teams. The solution architects worked separately from the team, using another building since they belong to the product management sector [1]. 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 [1]. On the other side, system architects also deal with features across several areas. They were closer to the teams in the building and their day-to-day activities [1]. The system architect was responsible for updating the backlog based on the teams' progress, arranging requirements, and designing workshops 4.5.7 [1]. **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 [1]. 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 that may arise with those features. Due to this, 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 [1].

4.5.17 Domain PO (1)

Name: domain PO. **Goal**: originally, LeSS [10] 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 [10]. **Who**: PO. **How**: in the four products evaluated in Uludag *et al.* [43] 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 [43]. 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 [43] 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 [43].

4.6 Scrum

The Scrum and its adaptation frameworks gathered the highest number of practices in our study within 32 different agile tailored practices. Being the most popular agile framework in the literature, 52 studies from the SLR were tailoring Scrum practices to accommodate their needs and uniqueness. Regarding the business domains of those studies, the Scrum tailored practices were seen in the automotive industry, BI and Big Data companies, broadcasting, enterprise CRMs, financial sector, general industry, healthcare organizations, internet, IT service providers, logistics, mission critical software, oil, and energy industry, process and industry automation, science and research, software service providers, and telecommunication.

4.6.1 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 [105]. 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 [105]. **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 [17–19, 24, 32, 50, 52, 56, 66, 68–70, 77, 78, 81, 84–86, 94, 96, 100]. 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 [19, 77, 86]. 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 [86]. 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, [77]. 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 [17–19, 31, 52, 56, 70]. Studies that involved onshore and offshore teams applied different strategies involving or not those teams [18, 50, 52, 81, 97]. 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 [52]. From another view, one study showed dailies only with the offshore team since the onshore team focused on generating and maintaining project specifications [81]. Some studies also reported the conduction of daily meetings through instant messages at chats or email [18, 69, 81, 84, 100] due to language problems [84]. In an exciting strategy, two case studies reported the presence of daily meetings specifically for testers and only to discuss test results [70, 94]. 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 [59]. The presence of two dailies was also common due to timezone differences in combining Indian, American, and European teams [31], or the number of people involved [56]. Also, two studies reported the presence of dailies every two days at least in one of their projects [56, 96]. Moreover, some papers reported the benefits of daily meetings on large-scale projects. First, it encouraged communication among the onshore and offshore teams [17, 69, 86]and helped them to communicate better and resolve issues faster [97]. 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 [2, 56], or teams involved in generating and maintaining project specifications [56]. **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 [24, 32, 66, 68, 78, 85]. However, very large-scale projects, such as the study presented by Lee and Yong [77] 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 meeting schedule due to the timezone differences or the number of people involved [19, 31, 56, 86]. 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 [18, 50, 52, 56, 81, 97]. Projects that involved the offshore teams in the daily have presented fewer issues regarding this activity [52]. Also, distributed teams that were small or working closer justified the execution of dailies through chat messages [18, 81]. 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 [18, 69, 81, 84, 100]. 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 [2, 56].

4.6.2 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 [7,8,10]. 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 [17, 19, 31, 45, 69, 77, 85, 94]. The team representatives must also align their teams' impediments generated for other teams or if they plan to do it [17, 19, 31, 45, 69]. 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 [18, 19, 31, 45, 46, 69], sometimes related to impediment metrics, weekly defects, overall sprint plan [46], and with the POs presence [31]. 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 [59]. Each meeting had its goals based on the audience, although the common three questions of Scrum were answered [59]. One study has reported a different use of the SoS meeting. First, it involves the leadership team, and the project manager manages it [63]. It was also held in an open space focusing on discussions regarding technical topics rather than dealing with impediments [63]. However, this approach did not go well during the time since the meeting lost its effectiveness and became a finger-pointing meeting [63]. Rolland *et al.* [92] conducted a case study of a large-scale agile project that involved 120 participants who 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 [70, 94]. **Context**: in the evaluated studies, very different contexts led the teams to apply and

tailor the SoS practice. In Gupta *et al.* [59] 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 [18, 19, 31, 45, 46, 69]. IT Service companies involved with different suppliers suffered similar challenges, although both studies only showed minimal tailoring passing the meetings to a digital channel [70, 94]. In studies of sectors using more traditional approaches, Gupta *et al.* [63] 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.* [92] 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 [77, 85].

4.6.3 Retrospective Meeting (10)

Name: retrospective meeting. **Goal**: according to the Scrum guide, the retrospective meeting aims to plan ways to increase quality and effectiveness [105]. 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 [105]. **Who**: Scrum Team, PO, and SM. **How**: many ways were chosen by the teams and organization to conduct retrospective meetings, including regular retrospective [2, 31, 62], common retrospectives with all the team [13, 17, 18, 69, 70, 94], retrospectives every second [56, 70, 94] or third sprint [57], and even the absence of the practice [56]. In Bass's case study [31], 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.* [2] 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.* [13] 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 [13]. In Hossain *et al.* [56], 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 [56]. 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 [56]. Then, those retrospectives started to be every second sprint when the operation began to run smoothly [56]. With a similar result, Lous *et al.* [57] has shown a case in which the retrospective meetings were held every third retrospective week. In Hoda *et al.* [62] 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.* [18, 69] 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 [18]. Finally, in Vallon *et al.* [70, 94] involving the same case, the common retrospective 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 [70, 94]. **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.* [70, 94] 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 [56, 57, 70, 94]. However, independently of the frequency, without the teams' engagement, the practice can be discontinued very quickly or segregated involving only SMs, and POs [56]. The team's maturity is also a criteria to consider in implementing the retrospective meeting. In Hoda *et al.* [62], 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 make them improve [2, 13].

4.6.4 Status Dashboard (10)

Name: Status Dashboard. **Goal**: Scrum Guide does not present a specific model for progress tracking through a dashboard, like Kanban [105]. 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 [13, 59, 61, 63–66, 70, 71, 94]. **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 [13, 64, 71]. 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 [64, 71]. In Helena *et al.* [13] 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 [13]. On the other side, many studies developed a Scrum Board, similar to some Kanban boards, but with different purposes. Gupta *et al.* [59] 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 [59]. Hoda *et al.* [61] 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 [61]. 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.* [65] 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.* [63] 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 [63]. In very different applicability, Lee *et al.* [66] 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 [66]. Finally, in Vallon *et al.* [70, 94] 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.* [66] study, apply a specific board just for issues tracking to map it and solve the issue quickly. In a similar environment, Gupta *et al.* [63, 65] 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 [63, 65]. 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 [61, 64, 71]. The studies that used physical dashboards had the privilege of having collocated teams among the distributed environment of their projects [13, 59]. However, this approach led the teams to share the dashboard through videoconference with remote teams [59] or to deal with issues in synchronizing physical and digital boards [13]. Further, the suppliers' relationship from *et al.* [70, 94] studies have presented the results of working in an environment with the feeling of "us and them".

4.6.5 Planning meeting (9)

Name: planning meeting. **Goal**: the planning meeting can be considered the starting point of what will be developed in a sprint [105]. 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 [105]. 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 [105]. **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 [17, 18, 56, 69, 70, 84, 94]. In their multiple-case study, Hossain *et al.* [56] 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 [56].

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 [56]. 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 [56]. In Paasivaara *et al.* [18, 69] 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 [18, 69]. 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 [18, 69]. From another perspective, in Vallon *et al.* [70, 94] 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 [70, 94]. 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 [70, 94]. In a simpler approach, Hole and Moe [84] 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-tasks, re-estimate, and validate them with the local teams [84]. 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 [84]. 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.* [17] 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 [17]. In a very different approach, Kussmaul [81] 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 [81]. Finally, in Scheerer and Kude's case study [93], 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. Companies with many teams spread around several countries were dividing their planning according to the overlap of hours between 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 [17, 18, 56, 69, 70, 84, 94]. 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 [18, 69]. Sometimes the meeting was even collocated, especially in the first planning 4.6.11, but just for the teams relatively close [18, 69]. 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 [81]. However, in cases where the remote or the offshore team was seen as less skilled compared to the onshore team, the remote team didn't have much power to discuss estimation and assignments, which can harm the agile development process [84]. Finally, more established sectors, such as the one presented by Paasivaara *et al.* [17] 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.

4.6.6 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 [105]. 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 [18–20, 52, 66, 69, 73, 83, 100]. In Lee *et al.* [66] 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 [66]. Paasivaara *et al.* [18, 69] 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.* [83] 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 [18–20, 52, 66, 69, 73, 83, 100]. Only two studies reported the presence of small-scale case projects, although the team members' distribution required the use of asynchronous tools [69, 100]. 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 [18–20, 52, 66, 73, 83].

4.6.7 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.* [63] 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 [45, 63, 82–84, 99]. Bass [45] 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 [45]. From another study perspective, Gupta *et al.* [63] 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 [63]. 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 [63]. Hossain *et al.* [83] 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 [105], Hole and Moe [84] reported a project manager responsible for assigning tasks to the team development team. Meanwhile, Cho [99] reported a project manager working on backlog refinement, breaking tasks that were constantly not completed. Moreover, in the Korhonen case study [82], 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.* [91] 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 [91]. Finally, Martini *et al.* [51] 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 [45, 83, 84] to the healthcare sector [63], and telecommunication sector [82] until mission-critical software organization [99]. In Gupta *et al.* [63] and Bass [45] 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.* [63] 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.* [83], 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 [84] 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 [99] 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 [82], Usman *et al.* [91] study, and Martini *et al.* [51] 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.

4.6.8 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 [105]. The PO and the stakeholders may

review what was accomplished and give feedback to the team [105]. 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 [13, 31, 47, 52, 53, 56, 69]. Further, due to the distributed environment, the demos were commonly held through video, and phone conferences using screen sharing [56, 69]. Bass [31], 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.* [47] 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 [47]. Then, the demo was specifically held with the system architect, a member of the APO role responsible for engaging with the team [47]. In Helena *et al.* [13] 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 [13]. In another study by Nyrud and Stray [52], the demo involved different areas, such as technical domain experts, test leaders, and business representatives. Moreover, in Hossain *et al.* [56], 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 [69] 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 [56, 69]. By the time the visits were possible, the demos were adapted to be held in the team spaces [69]. Nyrud and Stray [52] study was carried out in a company involved in the banking, insurance, and even pension sectors, 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 [52]. 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 [13, 31, 47, 53].

4.6.9 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 [73]. Scrum does not specify the use of wikis since the teams are collocated in the same physical space [105], 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 [18, 73, 77, 78, 81, 86, 99].**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.* [73] 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 [73]. In Prikladnicki and Wildt [78] 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 [78]. For a similar purpose, one of the companies from [18] used their backlog as a wiki, allowing everyone to access it and follow the project's progress. In the Cho [99] 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 [99]. 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 [81]. Further, in Dorairaj *et al.* [86] 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 [86]. Finally, Lee and Yong presented an experience report on Yahoo [77] that used the wiki to manage different backlogs from the teams and products. They also used images to emphasize the human element [77]. **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 [18, 77, 78]. In a similar context, the study of Dorairaj *et al.* [86] 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 use wikis for technical purposes focus 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 [73, 99]. 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 the Kussmaul [81] study, along with the project development, the frequency of emails was reduced, and the teams could control the mailing list.

4.6.10 Proxy Product Owner (PPO) (6)

Name: PPO. **Goal**: according to the Scrum guide [105], 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 [31, 32, 45, 47, 56, 87]. Due to the tailored nature, the role received different functions. **Who**: team members. **How**: in one of the Bass studies [31], 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 [31]. In another study, Paasivaara *et al.* [47] 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 [47]. Moreover, similar to a Scrum PO and a PPO, Hossain *et al.* [56] presented a proxy customer professional who checked the sprint progress and provided feedback to the team. In a similar perspective, but with a different purpose, Lehtinen *et al.* [87] 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 [87]. As an isolated customer representative, the POs did not participate in the regular Scrum events [87]. Finally, Bass studies [32, 45] 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 [31, 32, 45]. The author evaluated the tailored approaches used to accommodate the Scrum Master [31] and the Product Owner team [32, 45] 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.* [47] multiple-case study also focused on understanding the scaling of the PO role, the first case shows how the role was adapted to an APO, and the second case to the PPO model. Further, Hossain *et al.* [56] 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.* [87] study, could not be the best approach. The PO expectations failed in the study since they were not seen as part of the Scrum teams, and the developers did not know which PO was responsible for which requirements [87].

4.6.11 First collocated Sprint (6)

Name: first collocated Sprint. **Goal**: in the regular Scrum, a sprint represents a team iteration that should be committed to a goal [105]. At the end of a sprint, the team is expected to accomplish the goal as 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 [17, 26, 32, 69, 72, 86]. **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 [17, 26, 32, 69, 72, 86]. In Paasivaara and Lassenius [69] 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 [69]. In Bass [32] 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.* [86] 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 that, it would be natural to emerge the bonding between the entire team and the customer [86]. In Kommeren and Parviainen [72] experience report on Philips, the organization percevid the importance of gathering the teams physically, improving inter-team coordination performance. Moreover, Paasivaara *et al.* [17] study suggested collocating the teams in the initial iterations

and during critical phases of the project, such as major releases iterations. Finally, despite the importance of gathering customers and teams, Dorairaj *et al.* [26] 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 [17, 26, 32, 69, 72, 86]. In Paasivaara and Lassenius [69] 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 [17]. 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 [26, 32]. Finally, in teams specialized in developing electronic products, such as in the case of In Kommeren and Parviainen [72], gathering members from Asia and Europe helped them in the inter-team coordination activities of hardware products.

4.6.12 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 concerned individuals, process, DoD, interactions, and even tools [105]. **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.* [48] 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 [3] tool would start a new automated build. On this pipeline, the teams were using Ncover [4] to establish test code coverage beyond 80% since anything below it would block the build [48]. This approach ensured the quality of the code and monitored its progress without allowing quality loss. In Nyrud and Stray [52] study, the authors evaluated inter-coordination mechanisms. The case used Jira [5] 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 [52]. In another use of Jira, Paasivaara and Lassenius [69] 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 [85] 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 [99] that used VersionOne 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 [99]. **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 coordination and management tools. In the cases of Paasivaara and Lassenius [69] and Nyrud and Stray [52], 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.* [48], 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 [85] 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 [99], a more traditional tool such as VersionOne is more suitable.

4.6.13 Weekly status meeting (5)

Name: weekly status meeting. **Goal**: weekly status meeting is not described as a practice in the Scrum Guide [105]. However, large-scale projects needed to resolve issues, cross dependencies, and alignment gaps [46, 56, 59, 78, 79]. **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.* [46] 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

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

has also presented a global leadership meeting weekly focusing on reviewing overall program progress and key impediments [46]. In this meeting, the scope, schedule, cost, and quality status were reported to the senior management, who used this data to drive their decisions [46]. In an onshore and offshore relationship, Hossain *et al.* [56] 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 4.6.2. 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 this, 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 [59]. Further, in the Prikladnicki and Wildt [78] 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.* [79] presented a typical weekly meeting from a practitioner that took long periods to ease the resolution of challenges like communication gaps, team awareness, and morality issues. **Context**: Jha *et al.* [46] study was held at Siemens that used a hybrid development model combining traditional and agile practices. Due to this, 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 [46]. The onshore and offshore relationship seen in Hossain *et al.* [56] study required weekly meetings to accommodate the needs of the sites that required synchronization during the development. Gupta *et al.* [59] 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 [59]. Still, in line with traditional approaches, the Prikladnicki and Wildt [78] 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.* [79] used the weekly meeting to solve regular communication gaps and tech issues.

4.6.14 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 [105]. 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 [105]. **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 [59]. Similar to that checklist, Paasivaara *et al.* [18] study has shown a DoD checklist consisting of integration tests completed, version reports made, and user guides updated. From another study, Fitzgerald *et al.* [48] 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 [24] 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 this, 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 cases, code reviews, no defects, system tests, dependencies checks, and acceptance tests approval [24]. Finally, in Badampudi *et al.* [75] 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.* [48] 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 [24] 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 [24]. Meanwhile, the FDA-regulated environment from Badampudi *et al.* [75] 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 [18, 59].

4.6.15 Component Teams x Generalized teams (5)

Name: component teams x generalized teams. **Goal**: scrum teams should be cross-functional, which gathers all the necessary skills to create value constantly during sprints and achieve the product's goals [105]. They also should be self-managing teams and able to decide who implements what, when, and how [105]. 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 [47, 49, 74, 84, 88]. A component team is usually responsible for a component or part of the full product, having ownership over it and controlling the development of new features on it [47, 74, 84, 88]. 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 [49]. **Who**: Scrum Team. **How**: Martini and Bosch [49] 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 [49]. 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.* [47] 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 [84]. Further, Hole and Moe [84] presented a multiple-case study in which two 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 [84]. Sables *et al.* [88] 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 [88]. Finally, Koch *et al.* [74] 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 [74]. **Context**: Martini and Bosch's [49] 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 [49]. As expected from a generalized team, when the teams become responsible for the whole solution, they also lack proper knowledge of specific components that form the company portfolio [49]. 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.* [47]. However, experienced component teams have shown lower coordination work and higher expertise in coordinating activities than feature teams since they have a superficial knowledge of the project components [88]. 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 [74, 84].

4.6.16 Product Ownership (4)

Name: product ownership. **Goal**: product ownership is not a concept originally described in Scrum, but from Extreme Programming (XP) [106]. In XP, product ownership is usually held by the on-site customer, a client representative available for the team full-time [106]. 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 [105]. Meanwhile, product ownership can be described as a state that every Scrum team must chase to achieve. It means feeling like one of the owners of the product that understands 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 [50] involving 50 practitioners from 9 companies, the author percevid different configurations of which site the product ownership stayed. Sometimes, product ownership stays offshore, but it's more common on 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 [50]. Even when the development management stays offshore, the product ownership usually stays onshore [50]. Paasivaara *et al.* [47] study considered it difficult to implement shared responsibility through Product Ownership since the Proxy Product Owners 4.6.10 had specific product areas to work without getting involved in the whole product. Finally, Bass studies [32, 45] 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 showing product owners responsible for specific areas of the product or product parts [32, 45, 47]. 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 [32, 45, 47]. Meanwhile, Bass's study [50] shows different approaches to having product ownership onshore or offshore depending on customer knowledge, the company sector, and even management maturity.

4.6.17 Requirement Workshops (4)

Name: requirement workshop. **Goal**: design and requirement workshops are original from LeSS framework [10] and aim to help and clarify story aspects for the teams [10]. It was possible to identify some Scrum studies with the same purpose [47, 95, 97]. 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.* [47] 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 [47]. Whether needed, the requirement workshop could evolve into a design workshop for more detailed planning with the team responsible for the development [47]. In Noordeloos *et al.* [97] 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.* [95] 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 [95]. Finally, similar to the requirement workshop, Gupta *et al.* [63] 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 [63]. **Context**: the studies of Noordeloos *et al.* [97] and Daneva *et al.* [95] 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 [95, 97]. In Paasivaara *et al.* [47], the case that implemented requirement and design workshop used it as a communication technique for formal communication with the Scrum teams, improving collaboration among members and knowledge sharing. In a similar perception, the Gupta *et al.* [63] 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.

4.6.18 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 [105]. More than this, they must plan for the sprint, establish and apply the DoD definition, and stick to the sprint goal by adjusting their plan according to the necessary [105]. According to Scrum, any other functions related to the product, backlog, prioritization of backlog items, removal of impediments, and ensuring the Scrum events and coaching team members are the responsibilities of the PO and SM. However, a tailored approach was seen for those functions due to the distributed nature and the large-scale context. **Who**: developers. **How**: in Vallon *et al.* [70, 94], 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 [70, 94]. Due to this, the additional supplier suffered with alignments, process implementation, and even story discussion. Based on this, 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 [70, 94]. They became the unofficial Scrum master-like roles and improved the project's flow of information by taking care of the process implementation and discussion of impediments. In Hole and Moe's multiple case studies, [84], 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 [84]. However, a scrum master reported that discussions between sub-teams took too much time due to the distribution barrier. Due to this, the SM appointed one of the remote developers as a local SM, and the distributed SM mostly communicated with this person [84]. The approach reduced the long-time discussions and defined a focal point for discussions and process clarification for the team. Hoda *et al.* [61] conducted a large case study that presented an important scenario. In cases where the customer lacks involvement, the development teams saw the need to act as proxy product owners 4.6.10 to understand the customer needs better, and

the development team did it [61]. 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 [61]. **Context**: the Vallon *et al.* [70, 94] studies presented a very interesting scenario of suppliers' relationships through the development of large-scale distributed projects. Despite the additional team's skills, 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 [70, 94]. 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 [84]. 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 [84]. Finally, in typical relationships of consultancy firms with customers, some customers do not play an active role during the project [61]. Due to this, developers must understand the customer domain and process independently in Hoda *et al.* [61], or the project can suffer without progress.

4.6.19 Technical Debt Awareness (4)

Name: technical debt awareness. **Goal**: technical debts can be described as issues developed in the solution during development 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 [99]. Despite the origin, technical debts must be addressed and resolved along with the sprint [105] 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.* [95], 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 accumulating over time during development [95]. Consequently, new requirements may require architecture redesign to continue the development, which implicitly brings technical debt awareness to the table [95]. Further, in the Gupta *et al.* [59] 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 [59]. In this case study, the technical debts were considered stories, which may accomplish the criteria from the DoD 4.6.14, pass through code review, validate that the solution did not produce another technical debt, and prevent issues in the other areas of the solution [59]. Validation of technical debts occurred through automation tests in the workflows. Also, in this case, the organization built a dedicated multi-functional team focused on resolving technical debt stories, called TD team [59]. 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 debt backlog, the team was dissolved, and the members regrouped with other Scrum teams [59]. Further, in another Gupta *et al.* [63] 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 [63]. Finally, Sekitoleko *et al.* [76] 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 [76]. The managers, program officers, and the PO [76] identified the planned ones during the planning phase. Meanwhile, the unplanned would emerge while developing improper requirements [76]. **Context**: Daneva *et al.* [95] 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 [95]. 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 this, the current features can constantly impact the solution [95]. In Gupta *et al.* [59] 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 convince managers to include the debts in the iterations. Also, in test case development and automation [59]. 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.*, [63], the event day for technical debts in the healthcare sector focused specifically on the team pain areas that would improve product quality. Further, the case of Sekitoleko *et al.* [76] 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 [76].

4.6.20 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 [105]. The stakeholder and the PO will inspect the outcome of the Sprint and determine future adaptations [105]. The Product Backlog may also be adjusted according to the progress achieved [105]. 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 [47, 56, 70, 94]. **Who**: Scrum team, Product Owner, Scrum Master, stakeholders, customers. **How**: in Vallon *et al.* [70, 94] 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 [70, 94]. Additionally, one of the SMs from the additional supplier is present at the main site while the rest of the team observes the meeting [70, 94]. During the review, the teams should demonstrate developed features and discuss the current product increments [70, 94]. In Paasivaara *et al.* [47] 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.* [56], a 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 [56]. In another company in the telecommunication sector, a joint sprint review is held at the end of each sprint. However, only the management team participated during the customer demo, PO, SM, and project manager, instead of the whole Scrum team [56]. 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 [56]. 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 [56]. Finally, in the industry case, the developed code base was released to the onshore test engineer instead of a formal review meeting for acceptance testing [56]. **Context**: Vallon *et al.* [70, 94] studies involving different suppliers had a review meeting that contributed to the presence of another tailored practice, the Developers as Scrum Masters 4.6.18. 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 guarantee their teams' presence just for observation [70, 94]. In Paasivaara *et al.* [47], 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.* [56] 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 less traditional, the review meeting demo process involved only the management team, disconsidering the importance of involving all members from the Scrum team [56]. By the way, the authors did not describe the perception of the team and the consequence of not involving them. 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 [56]. However, those companies are IT service providers and an industry that represents different sectors that used the review meeting as an activity of the quality team [56].

4.6.21 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 [105]. Based on a similar purpose, some studies established a maintenance team to handle all the requests regarding the product operation [17, 19, 69]. **Who**: Scrum Team. **How**: Paasivaara *et al.* [17, 19] 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 [17, 19]. All customers could add new issues to the maintenance backlog through Jira. However, the organization checked the issue initially, passing it to the specific product, and the PO would contact the maintenance team after a quick verification of the issue [17, 19]. 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 [17, 19]. 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 [17, 19]. In Paasivaara and Lassenius [69] 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 split the business units' demands better. The customer, the maintenance team, stands out as a suitable option. In both studies from Paasivaara *et al.* [17, 19, 69], 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.

4.6.22 Technical Area Responsible (TAR) (3)

Name: TAR. **Goal**: Scrum does not describe a specific area responsible for the technical subjects of the team [105]. 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.* [54] 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 who know more about the project and technologies used in the project [54]. The TARs were 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 [54]. At Ericsson, the TARs had more responsibilities, like code review, identifying quality issues or improvements for the POs, rejecting design proposals, developing guidelines, and prioritizing trouble reports [54]. Nyrud and Stray [52] 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 possessing technical insights into the entire product portfolio and serves as a link between the different teams [52]. 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 [52]. Finally, Helena *et al.* [13] has described the presence of specific technical leads for the testers and the developers. Different technical leaders managed each group. **Context**: Moe *et al.* [54] 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 [54]. Nyrud and Stray [52] 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 [52]. Finally, Helena *et al.* [13] conducted a case study on BBC with 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.

4.6.23 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, harming the project budget. To solve that kind of issue, some companies started to work with buffered fixed-bid contracts [17, 61] or iteration contracts [98]. **Who**: the organization, management. **How**: Hoda *et al.* [61] 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 to develop the project or the feature [61]. Based on it, the contract is drawn on the estimate considering the buffer for a fixed price and scope [61]. 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.* [17]. 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 [61]. By doing this, the organizations were selling an agile trial basis for the customers, which helped them build confidence with the customer and reduce risk [61]. 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 [61]. Finally, Batra *et al.* [98] 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 [98]. **Context**: The tailoring strategy of fixed-bid contracts from Hoda *et al.* [61] 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 this, the organization must map agile practice into customer practices [61]. Based on the customer orders, the organization used the agile data to estimate it, like team size, velocity, burndown chart, and members turnover [61]. From another approach, some practitioners interviewed by Hoda *et al.* [61] 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 [61]. Further, allowing customers to swap features shows them that the supplier can change according to the Agile manifesto [15]. Still, without losing control of the estimated effort, [61]. In the Paasivaara *et al.* [17] 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.* [98] consist of a theoretical framework developed based on the literature. Still, real experience is much more advantageous in those cases regarding contracts and costs.

4.6.24 Code freeze (3)

Name: code freeze. **Goal**: code freeze is not a technique specified in the Scrum guide [105], 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 [20, 31, 67]. **Who**: Scrum Team, management. **How**: to avoid any new updates ahead of customers' demonstrations, one of the practitioners on Bass study [31] reported using 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 [20], one practitioner reported using code freeze before the start of regression tests. Due to this, 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 [20]. However, this process was handled by a release department, not the particular teams [20]. Finally, Laukkanen *et al.* [67] 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 had to freeze the source code and could develop only critical bug fixes afterward, but in practice, the code freeze was not respected [67]. 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 [67]. 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, [31], the practitioner who 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, [20], the product company that reported the three day 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.* [67] 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 was big enough for the teams to ignore the practice rule.

4.6.25 Community of Practice (CoP) (3)

Name: Community of Practice (CoP). **Goal**: the community of practice (CoP) is very common in other agile frameworks [7, 10]. However, it is not originally described on Scrum [105]. 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 [54, 59, 63]. **Who**: Scrum Team, project manager. **How**: Moe *et al.* [54] case study at Ericsson involving Swedish and Chinese teams applied tech forums, similar to CoPs. Those CoPs included test, integration, development, and SMs forums [54]. In Gupta *et al.* [59] experience report, the Scrum team testers dealt with many responsibilities, from test automation to developer support. Beyond that, they were responsible for conducting test CoPs among the members [59]. Finally, in another experience report from Gupta *et al.*, [63], 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 [63]. **Context**: Gupta *et al.* [63] experience report from a healthcare company has presented 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 [63]. 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 [1, 43]. In Ericsson's case study from Moe *et al.*, [54], 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 [54]. Finally, the other experience report from Gupta *et al.* [59] 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.

4.6.26 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 [105]. They are also responsible for leading, training, and coaching the organization in its Scrum adoption [105]. In large-scale distributed projects, agile coaching is still required for more adoption of Scrum [74, 77, 90]. **Who**: Scrum Team, SM, PO, external consultant. **How**: in one of the cases from Koch *et al.* [74], 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 [90] 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 [90] in the project. Finally, Lee and Yong presented an experience report on Yahoo [77], showing that global product and international teams received early coaching and training from the corporate Agile group. However, the international ones were unfamiliar initially [77]. **Context**: Lee and Yong presented an experience report on Yahoo [77] 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 [90] 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 essential factors for success. [90]. Finally, in Koch *et al.* [74] 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 [74].

4.6.27 Area Product Owner (APO) (2)

Name: APO. **Goal**: an APO is not described in any Scrum documents or the Scrum Guide [105]. However, it is a common practice from the fewer framework [10], in which an Area Product Owner focuses on a customer-centric area and acts as PO concerning the teams of that area. Even though it is an original practice from another framework, it was perceived in two Scrum studies. **Who**: PO. **How**: Paasivaara *et al.* [47] 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 an APO headed each area [47]. Like LeSS [10], the APO manages an area-specific backlog, and together with the PO, they formed the PO teams [47]. The APO is supposed to work with 2-3 teams developing and managing features of one specific product [47]. In another study from Moe *et al.* [54], 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 [54]. **Context**: Moe *et al.* [54] study was conducted in a very traditional company with 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 [54]. Further, in Paasivaara *et al.* [47] 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 [107]. Due to the project's needs, APO's role was divided between a system architect and a solution architect [47].

4.6.28 Behavior Driven Development (BDD) (2)

Name: BDD. **Goal**: BDD is a not a original Scrum practice [105]. 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 [70, 94]. 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.* [70, 94] 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 using BDD in the same code base. 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 [70, 94]. 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 [70, 94]. Moreover, testers constantly struggled to finish the automation of BDD scenarios within the sprints, which resulted in issues during delivery [70, 94]. **Context**: Vallon *et al.* [70, 94] 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 test cases that had the stories developed [70, 94]. 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.

4.6.29 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 [105] or even any other agile framework. The design pipeline aims to establish a design sprint ahead of the development sprint [61, 66]. 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.* [66] 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.* [61] 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 [61]. 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 [61]. 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 [61]. **Context**: Lee *et al.* [66] 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.* [61] 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.

4.6.30 Futurospective (1)

Name: futurospective meeting. **Goal**: futurospective is not nearly an original practice from Scrum [105]. 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 [25]. It is a 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.*, [25], based on the question, the coaches and managers started to write a "showcase" [25]. 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.* [25] conducted a case study at Ericsson to describe how a new organization acquired by Ericsson used "Value Workshop" the different sites and teams that are 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 originated from their interpretation and behavioral implications as a team from the workshops and futurospective meetings [25].

4.6.31 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 [105]. The PO also needs to be the face and voice of the product since the role represents the needs of the customer, stakeholders, and end-users [105]. However, story owner is a tailored practice to substitute the regular PO role from Scrum [61]. Instead of being responsible for the whole product, they are responsible for particular stories of less than a week long [61]. **Who**: story owners. **How**: the concept of story owner was seen in the cases from Hoda *et al.* [61]. 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 [61]. 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 [61]. Such a role becomes more present when customer involvement is insufficient for the project, and the supplier or team members cannot change this nature [61]. The approach also allowed the teams to plan new stories based on the story being developed with the story owner. It also allowed the customer representatives to create a sense of ownership among the stories they were responsible for [61]. **Context**: Hoda *et al.* [61] 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 [61]. 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.

4.6.32 Limited blast radius technique (1)

Name: limited blast radius technique.**Goal**: this practice originates itself in some studies that used the Spotify framework [27]. However, with the same goal of releasing a new version of the solution to a small portion of end-users [20] 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 [7], one of the companies from Bass study [20] 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 [20]. 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 [20] without harming the entire customer community. **Context**: in the case study presented by Bass with seven international companies [20], 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.

5 Discussion

Agile practices have become increasingly popular in the industry for coordinating development efforts, particularly for large-scale projects distributed among different teams [13, 17]. However, it's essential to recognize that simply using remote tools to compensate for the lack of physical interaction is insufficient to tailor agile practices for these projects. The team's unique context must be considered, including factors like size, location, time zone differences, regulatory compliance, technical complexity, and the company's history and discipline. These contextual elements can significantly impact the team's agile tailoring journey [6, 60].

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 can cover 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, agile frameworks have benefited DSD teams working on large-scale projects. Studies have shown that the frameworks [3, 18, 23, 27, 43, 55, 56, 60] have aided in achieving project goals. However, these frameworks often require customization to fit the teams' specific needs. For instance, the consultancy firm in Pandya's report [28] used specific tailoring techniques to carry out SAFe transformation. Similarly, Paasivaara conducted a case study evaluating the adoption of SAFe at an industrial company with two business lines [23]. Both studies required different tailoring approaches to meet the teams' needs. For example, the RTE in Paasivaara's study worked part-time on other activities, while in Pandya's report, the RTE had closer responsibilities to what is defined in SAFe.

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 those case contexts, we reinforce the belief of Ambler and Lines that most of the several frameworks [7–12] 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 provides a context-specific guide to help organizations according to their uniqueness during the agile tailoring process. Considering the context aspects of the teams, this study presents a collection of practices and different ways of applying based on different teams' necessities.

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 [89]. The point here is that Spotify [7], SAFe [8], and LeSS [10] 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.

From 96 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 4.2.6 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 4.3.8. 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 4.4.3. Moreover, in LeSS, we could see the addition of the DoD definition, including the DoE and the DoR 4.5.9. The DoE describes the criteria for rough requirements to be broken into individual stories. In contrast, DoR describes the criteria for which the user stories must 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 4.6.14.

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 4.2.11, 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 4.3.10, Less section 4.5.11, and Scrum 4.6.2).

This study's results answer two key research questions by identifying and describing how DSD teams apply agile frameworks in large-scale settings. Through a Systematic Literature Review (SLR), we have identified and described 96 tailored agile practices used by distributed teams applying agile frameworks on a large scale. We hope to provide a context-specific guide to help organizations with their unique needs during agile tailoring. The study's results offer a collection of tailored agile practices and describe various ways of applying them based on real industry cases from the literature, taking into account the context aspects of the teams and their business domains, the scale dimension, and the agile framework in use.

The SLR has identified mature agile tailoring studies that use empirical research methods with high rigor. Although some studies are based on experience reports without empirical evidence, they still support their claims. Evaluation research is their primary focus, with most practices coming from unstructured sources evaluated by experts or surveys. The present SLR provides an organized view of agile method tailoring techniques adopted by organizations, enhancing the maturity of the research topic in the literature.

Finally, our SLR studies on agile tailoring started in 2007, focusing on distributed teams in large-scale settings. This timeframe allows for technological advancements to support remote work globally and increased

research on this topic since the 2010s with the emergence of agile and scaling frameworks. Previous SLR studies cover material dating back to 2002, as shown in Campanelli's 2015 study [6].

5.1 Limitations

Although the SLR was developed according to the guidelines outlined by Kitchenham and Stuart [34], it does have some limitations.

For ensuring construct validity, which investigates 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 [38]. The evaluations of the SLR studies were conducted using a peer-reviewed approach involving all the authors through the data extraction process regarding the tailored agile practices. All authors were involved during the selection and description of agile tailored practices, and if any of them disagreed with the practice, we would discuss it to reach a consensus.

Regarding external validity [38], which is related to what extent it is possible to generalize the findings and to what extent they have value to practitioners and researchers [38]. we conducted the SLR using a research protocol developed and validated by the authors. In addition, we chose to search through highly respected databases for research on agile and distributed development, including ACM, IEEE Xplore, Springer, Scopus, and Wiley. Using a pre-defined search string in these bibliographic databases, which are well-known for agile development references, we obtained comprehensive and generalized findings, as most studies in the field are published in these 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 [99]. However, it is important to point out that the absence of the snowballing technique [108] in the SLR may increase the external validity threat.

For conclusion validity, which is concerned with to what extent the data and the analysis are consistent [38]. We understand that using fewer bibliographic databases may have made the research lose relative papers, but choosing the most renowned ones could provide broader coverage. However, replicating the process of the SLR can lead to different results due to the difference in individual decisions and evaluation of the linked practices among researchers.

Finally, to ensure internal validity, concerned with the treatments' effects on the variables due to uncontrolled factors in the environment [38]. Following the research protocol, we aimed to mitigate this threat by carefully analyzing the studies, results, and redundancy. Besides, every study was chosen, and the authors discussed the agile tailored practices extracted from them. When one disagreed with a choice, we all discussed it to reach a common agreement. Moreover, repeating the research protocol using the same search engines can ease the reproduction of the 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 particularities and needs. Due to this, 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 96 tailored practices were mapped based on the literature. In addition, the present study has also shown how the 96 customized practices were tailored and implemented in several different contexts. For future research, we'll use insights from our SLR to create a framework, test it in industry, and continue researching to improve it.

This study analyzes how agile tailoring works in DSD teams using agile and scaling agile frameworks in large-scale settings. It provides valuable insights for academia and industry and offers tailored approaches for researchers and practitioners to understand how different circumstances require different techniques. For future research, we look forward to developing a framework based on the SLR findings, checking the presence of those practices in a real industrial case, and continuing the DLR in the following years to expand the findings and keep it updated.

References

[1] M. Paasivaara and C. Lassenius, "Scaling scrum in a large globally distributed organization: A case study," in *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 2016, pp. 74–83.

[2] R. Vallon, C. Dräger, A. Zapletal, and T. Grechenig, "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*, 2014, pp. 52–60.

[3] M. A. Razzak, I. Richardson, J. Noll, C. N. Canna, and S. Beecham, "Scaling agile across the global organization: An early stage industrial safe self-assessment," in *Proceedings of the 13th International Conference on Global Software Engineering*, ser. ICGSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 121–130. [Online]. Available: https://doi.org/10.1145/3196369.3196373

[4] R. Camara, A. Alves, I. Monte, and M. Marinho, "Agile global software development: A systematic literature review," in *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, ser. SBES '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 31–40. [Online]. Available: https://doi.org/10.1145/3422392.3422411

[5] VersionOne, Inc., "16th Annual State of Agile Development Survey," https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf, 2023, [Online; accessed 15-March-2023].

[6] A. S. Campanelli and F. S. Parreiras, "Agile methods tailoring – a systematic literature review," *Journal of Systems and Software*, vol. 110, pp. 85–100, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121215001843

[7] K. Henrik and I. Anders, "Scaling agile @ spotify with tribes, squads, chapters & guilds," 2012.

[8] Leffingwell, Dean, "Scaled Agile Framework 6.0," *https://scaledagileframework.com*, 2023, [Online; accessed 15-March-2023].

[9] S. W. Ambler and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*, 1st ed. IBM Press, 2012.

[10] C. Larman and B. Vodde, *Large-scale Scrum: More with LeSS*, ser. A Mike Cohn signature book. Addison-Wesley, 2016. [Online]. Available: https://books.google.com.br/books?id=oZ_vngEACAAJ

[11] Sutherland, Jeff and Brown, Alex, "Scrum@Scale," https://www.scrumatscale.com/scrum-at-scale-guide/, 2021, [Online; accessed 16-July-2022].

[12] Schwaber, Ken and Sutherland, Jeff, "The Scrum Guide," *https://www.scrum.org/resources/scrum-guide*, 2022, [Online; accessed 06-August-2022].

[13] H. Tendedez, M. A. M. Ferrario, and J. Whittle, "Software development and cscw: Standardization and flexibility in large-scale agile development," *Proc. ACM Hum. Comput. Interact.*, vol. 2, no. CSCW, nov 2018. [Online]. Available: https://doi.org/10.1145/3274440

[14] M. Lines and S. Ambler, *Choose Your WoW!: A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working (WoW)*, ser. Choose Your WoW Series. Independently Published, 2019. [Online]. Available: https://books.google.com.br/books?id=gTd4wAEACAAJ

[15] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," 2001. [Online]. Available: http://www.agilemanifesto.org/

[16] M. Paasivaara and C. Lassenius, "Communities of practice in a large distributed agile software development organization-case ericsson," *Information and Software Technology*, vol. 56, no. 12, pp. 1556 – 1577, 2014, special issue: Human Factors in Software Development. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584914001475

[17] M. Paasivaara, S. Durasiewicz, and C. Lassenius, "Using scrum in distributed agile development: A multiple case study," in *2009 Fourth IEEE International Conference on Global Software Engineering*, 2009, pp. 195–204.

[18] ——, "Using scrum in distributed agile development: A multiple case study," in *2009 Fourth IEEE International Conference on Global Software Engineering*, 2009, pp. 195–204.

[19] ——, "Distributed agile development: Using scrum in a large project," in *2008 IEEE International Conference on Global Software Engineering*, 2008, pp. 87–95.

[20] J. M. Bass, "Influences on agile practice tailoring in enterprise software development," in *2012 Agile India*, 2012, pp. 1–9.

[21] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *Journal of Systems and Software*, vol. 119, pp. 87–108, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121216300826

[22] T. Dingsøyr, T. E. Fægri, and J. Itkonen, "What is large in large-scale? a taxonomy of scale for agile software development," in *PROFES*, 2014.

[23] M. Paasivaara, "Adopting safe to scale agile in a globally distributed organization," in *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 2017, pp. 36–40.

[24] S. Matthiesen and P. Bjørn, "When distribution of tasks and skills are fundamentally problematic: A failure story from global software outsourcing," *Proc. ACM Hum.-Comput. Interact.*, vol. 1, no. CSCW, dec 2017. [Online]. Available: https://doi.org/10.1145/3139336

[25] M. Paasivaara, O. Väättänen, M. Hallikainen, and C. Lassenius, "Supporting a large-scale lean and agile transformation by defining common values," in *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*, T. Dingsøyr, N. B. Moe, R. Tonelli, S. Counsell, C. Gencel, and K. Petersen, Eds. Cham: Springer International Publishing, 2014, pp. 73–82.

[26] S. Dorairaj, J. Noble, and G. Allan, "Agile software development with distributed teams: Senior management support," in *2013 IEEE 8th International Conference on Global Software Engineering*, 2013, pp. 197–205.

[27] A. Salameh and J. Bass, "Spotify tailoring for b2b product development," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2019, pp. 61–65.

[28] A. Pandya, V. S. Mani, and A. Pattanayak, "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*, ser. ICGSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–5. [Online]. Available: https://doi.org/10.1145/3372787.3390441

[29] K. Conboy and B. Fitzgerald, "Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 1, jul 2010. [Online]. Available: https://doi.org/10.1145/1767751.1767753

[30] B. Fitzgerald, N. L. Russo, and T. O'Kane, "An empirical study of system development method tailoring in practice," in *European Conference on Information Systems (ECIS)*, 01 2000, pp. 187–194. [Online]. Available: https://api.semanticscholar.org/CorpusID:7540031

[31] J. M. Bass, "Scrum master activities: Process tailoring in large enterprise projects," in *2014 IEEE 9th International Conference on Global Software Engineering*, 2014, pp. 6–15.

[32] ——, "How product owner teams scale agile methods to large distributed enterprises," *Empirical Softw. Engg.*, vol. 20, no. 6, p. 1525–1557, dec 2015. [Online]. Available: https://doi.org/10.1007/s10664-014-9322-z

[33] M. Alqudah and R. Razali, "A review of scaling agile methods in large software development," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, pp. 828–837, 2016.

[34] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Technical report, EBSE Technical Report EBSE-2007-01, Tech. Rep., 2007.

[35] A. B. Zamboni, A. D. Thommazo, E. C. M. Hernandes, and S. C. P. F. Fabbri, "Start uma ferramenta computacional de apoio à revisão sistemática," in *Brazilian Conference on Software: Theory and Practice - Tools session*, 2010.

[36] T. Dyba, T. Dingsoyr, and G. K. Hanssen, "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, pp. 225–234.

[37] M. Ivarsson and T. Gorschek, "A method for evaluating rigor and industrial relevance of technology evaluations," *Empirical Software Engineering*, vol. 16, pp. 365–395, 06 2011.

[38] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012, pp. 123–151.

[39] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *2011 International Symposium on Empirical Software Engineering and Measurement*, 2011, pp. 275–284.

[40] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE'08. Swindon, GBR: BCS Learning & Development Ltd., 2008, p. 68–77.

[41] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: A proposal and a discussion," *Requir. Eng.*, vol. 11, pp. 102–107, 03 2006.

[42] B. Glaser, *Emergence Vs Forcing: Basics of Grounded Theory Analysis*, ser. Emergence vs. forcing. Sociology Press, 1992. [Online]. Available: https://books.google.com.br/books?id=1ZJiQgAACAAJ

[43] O. Uludag, M. Kleehaus, N. Dreymann, C. Kabelin, and F. Matthes, "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)*, 2019, pp. 22–29.

[44] C. Ebert and M. Paasivaara, "Scaling agile," *IEEE Software*, vol. 34, no. 6, pp. 98–103, 2017.

[45] J. M. Bass, "Agile method tailoring in distributed enterprises: Product owner teams," in *2013 IEEE 8th International Conference on Global Software Engineering*, 2013, pp. 154–163.

[46] M. M. Jha, R. M. F. Vilardell, and J. Narayan, "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)*, 2016, pp. 84–88.

[47] M. Paasivaara, V. T. Heikkilä, and C. Lassenius, "Experiences in scaling the product owner role in large-scale globally distributed scrum," in *2012 IEEE Seventh International Conference on Global Software Engineering*, 2012, pp. 174–178.

[48] B. Fitzgerald, K.-J. Stol, R. O'Sullivan, and D. O'Brien, "Scaling agile methods to regulated environments: An industry case study," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 863–872.

[49] A. Martini and J. Bosch, "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)*, 2016, pp. 1–10.

[50] J. M. Bass, "Large-scale offshore agile tailoring: Exploring product and service organisations," in *Proceedings of the Scientific Workshop Proceedings of XP2016*, ser. XP '16 Workshops. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2962695.2962703

[51] A. Martini, L. Pareto, and J. Bosch, "Communication factors for speed and reuse in large-scale agile software development," in *Proceedings of the 17th International Software Product Line Conference*, ser. SPLC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 42–51. [Online]. Available: https://doi.org/10.1145/2491627.2491642

[52] H. Nyrud and V. Stray, "Inter-team coordination mechanisms in large-scale agile," in *Proceedings of the XP2017 Scientific Workshops*, ser. XP '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3120459.3120476

[53] K. H. Rolland, "Scaling across knowledge boundaries: A case study of a large-scale agile software development project," in *Proceedings of the Scientific Workshop Proceedings of XP2016*, ser. XP '16 Workshops. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2962695.2962700

[54] N. B. Moe, D. Šmite, A. Šundefinedblis, A.-L. Börjesson, and P. Andréasson, "Networking in a large-scale distributed agile project," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2652524.2652584

[55] A. Salameh and J. M. Bass, "Heterogeneous tailoring approach using the spotify model," in *Proceedings of the Evaluation and Assessment in Software Engineering*, ser. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 293–298. [Online]. Available: https://doi.org/10.1145/3383219.3383251

[56] E. Hossain, P. L. Bannerman, and R. Jeffery, "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*, ser. ICSSP '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 110–119. [Online]. Available: https://doi.org/10.1145/1987875.1987894

[57] P. Lous, P. Tell, C. B. Michelsen, Y. Dittrich, M. Kuhrmann, and A. Ebdrup, "Virtual by design: How a work environment can support agile distributed software development," in *Proceedings of the 13th International Conference on Global Software Engineering*, ser. ICGSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 102–111. [Online]. Available: https://doi.org/10.1145/3196369.3196374

[58] R. Lal and T. Clear, "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*, ser. ICGSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 59–68. [Online]. Available: https://doi.org/10.1145/3196369.3196378

[59] R. K. Gupta, P. Manikreddy, and K. C. Arya, "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*, ser. ISEC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 147–156. [Online]. Available: https://doi.org/10.1145/3021460.3021478

[60] A. W. Brown, S. Ambler, and W. Royce, "Agility at scale: Economic governance, measured improvement, and disciplined delivery," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13. IEEE Press, 2013, p. 873–881.

[61] R. Hoda, P. Kruchten, J. Noble, and S. Marshall, "Agility in context," *SIGPLAN Not.*, vol. 45, no. 10, p. 74–88, oct 2010. [Online]. Available: https://doi.org/10.1145/1932682.1869467

[62] R. Hoda and J. Noble, "Becoming agile: A grounded theory of agile transitions in practice," in *Proceedings of the 39th International Conference on Software Engineering*, ser. ICSE '17. IEEE Press, 2017, p. 141–151. [Online]. Available: https://doi.org/10.1109/ICSE.2017.21

[63] R. K. Gupta, S. Jain, and B. Singh, "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)*, ser. ICGSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 77–81. [Online]. Available: https://doi.org/10.1145/3196369.3196389

[64] C. P. Godoy, L. M. Santos, A. F. Cruz, R. S. Zerbini, E. P. Silva, and C. A. L. Pahins, "Blueprint model: A new approach to scrum agile methodology," in *Proceedings of the 14th International Conference on Global Software Engineering*, ser. ICGSE '19. IEEE Press, 2019, p. 85–89. [Online]. Available: https://doi.org/10.1109/ICGSE.2019.00014

[65] R. K. Gupta, M. Venkatachalapathy, and F. K. Jeberla, "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)*, 2019, pp. 30–34.

[66] J. C. Lee, T. K. Judge, and D. S. McCrickard, "Evaluating extreme scenario-based design in a distributed agile team," in *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 863–877. [Online]. Available: https://doi.org/10.1145/1979742.1979681

[67] E. Laukkanen, T. O. Lehtinen, J. Itkonen, M. Paasivaara, and C. Lassenius, "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*, ser. ESEM '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2961111.2962608

[68] J. Garbajosa, A. Yagüe, and E. Gonzalez, "Communication in agile global software development: An exploratory study," in *On the Move to Meaningful Internet Systems: OTM 2014 Workshops*, R. Meersman, H. Panetto, A. Mishra, R. Valencia-García, A. L. Soares, I. Ciuciu, F. Ferri, G. Weichhart, T. Moser, M. Bezzi, and H. Chan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 408–417.

[69] M. Paasivaara and C. Lassenius, *Using Scrum Practices in GSD Projects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 259–278. [Online]. Available: https://doi.org/10.1007/978-3-642-12442-6_17

[70] R. Vallon, S. Strobl, M. Bernhart, and T. Grechenig, "Inter-organizational co-development with scrum: Experiences and lessons learned from a distributed corporate development environment," in *Agile Processes in Software Engineering and Extreme Programming*, H. Baumeister and B. Weber, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 150–164.

[71] S. Rahy and J. Bass, "Information flows at inter-team boundaries in agile information systems development," in *Information Systems*, M. Themistocleous and P. Rupino da Cunha, Eds. Cham: Springer International Publishing, 2019, pp. 489–502.

[72] R. Kommeren and P. Parviainen, "Philips experiences in global distributed software development," *Empirical Softw. Engg.*, vol. 12, no. 6, p. 647–660, dec 2007. [Online]. Available: https://doi.org/10.1007/s10664-007-9047-3

[73] M. Korkala, M. Pikkarainen, and K. Conboy, "Distributed agile development: A case study of customer communication challenges," in *Agile Processes in Software Engineering and Extreme Programming*, P. Abrahamsson, M. Marchesi, and F. Maurer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 161–167.

[74] C. Koch, C. Jørgensen, M. Olsen, and T. Tambo, "We all know how, don't we? on the role of scrum in it-offshoring," in *Creating Value for All Through IT*, B. Bergvall-Kåreborn and P. A. Nielsen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 96–112.

[75] D. Badampudi, S. A. Fricker, and A. M. Moreno, "Perspectives on productivity and delays in large-scale agile projects," in *Agile Processes in Software Engineering and Extreme Programming*, H. Baumeister and B. Weber, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 180–194.

[76] N. Sekitoleko, F. Evbota, E. Knauss, A. Sandberg, M. Chaudron, and H. H. Olsson, "Technical dependency challenges in large-scale agile software development," in *Agile Processes in Software Engineering and Extreme Programming*, G. Cantone and M. Marchesi, Eds. Cham: Springer International Publishing, 2014, pp. 46–61.

[77] S. Lee and H.-S. Yong, "Distributed agile: Project management in a global environment," *Empirical Software Engineering*, vol. 15, pp. 204–217, 04 2010.

[78] D. Wildt and R. Prikladnicki, *Transitioning from Distributed and Traditional to Distributed and Agile: An Experience Report*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 31–46. [Online]. Available: https://doi.org/10.1007/978-3-642-12442-6_3

[79] S. S. Hossain, "Challenges and mitigation strategies in reusing requirements in large-scale distributed agile software development: A survey result," in *Intelligent Computing*, K. Arai, R. Bhatia, and S. Kapoor, Eds. Cham: Springer International Publishing, 2019, pp. 920–935.

[80] T. Lautert, A. G. S. S. Neto, and N. P. Kozievitch, "A survey on agile practices and challenges of a global software development team," in *Agile Methods*, P. Meirelles, M. A. Nelson, and C. Rocha, Eds. Cham: Springer International Publishing, 2019, pp. 128–143.

[81] C. Kussmaul, *Onshore and Offshore Outsourcing with Agility: Lessons Learned*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 91–106. [Online]. Available: https://doi.org/10.1007/978-3-642-12442-6_6

[82] K. Korhonen, "Migrating defect management from waterfall to agile software development in a large-scale multi-site organization: A case study," in *Agile Processes in Software Engineering and Extreme Programming*, P. Abrahamsson, M. Marchesi, and F. Maurer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 73–82.

[83] E. Hossain, M. Ali Babar, and J. Verner, "Towards a framework for using agile approaches in global software development," in *Product-Focused Software Process Improvement*, F. Bomarius, M. Oivo, P. Jaring, and P. Abrahamsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 126–140.

[84] S. Hole and N. B. Moe, "A case study of coordination in distributed agile software development," in *Software Process Improvement*, R. V. O'Connor, N. Baddoo, K. Smolander, and R. Messnarz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 189–200.

[85] A. Välimäki and J. Kääriäinen, "Patterns for distributed scrum — a case study," in *Enterprise Interoperability III*, K. Mertins, R. Ruggaber, K. Popplewell, and X. Xu, Eds. London: Springer London, 2008, pp. 85–97.

[86] S. Dorairaj, J. Noble, and P. Malik, "Understanding team dynamics in distributed agile software development," in *Agile Processes in Software Engineering and Extreme Programming*, C. Wohlin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 47–61.

[87] T. O. Lehtinen, R. Virtanen, V. T. Heikkilä, and J. Itkonen, "Why the development outcome does not meet the product owners' expectations?" in *Agile Processes in Software Engineering and Extreme Programming*, C. Lassenius, T. Dingsøyr, and M. Paasivaara, Eds. Cham: Springer International Publishing, 2015, pp. 93–104.

[88] A. Sablis, D. Smite, and N. Moe, "Team-external coordination in large-scale software development projects," *J. Softw. Evol. Process*, vol. 33, no. 3, mar 2021. [Online]. Available: https://doi.org/10.1002/smr.2297

[89] S. Beecham, T. Clear, R. Lal, and J. Noll, "Do scaling agile frameworks address global software development risks? an empirical study," *Journal of Systems and Software*, vol. 171, p. 110823, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220302181

[90] B. Hobbs and Y. Petit, "Agile methods on large projects in large organizations," *Project Management Journal*, vol. 48, pp. 3–19, 06 2017.

[91] M. Usman, R. Britto, L.-O. Damm, and J. Börstler, "Effort estimation in large-scale software development: An industrial case study," *Information and Software Technology*, vol. 99, pp. 21–40, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584918300326

[92] K. H. Rolland, B. Fitzgerald, T. Dingsøyr, and K.-J. Stol, "Problematizing agile in the large: Alternative assumptions for large-scale agile development," in *ICIS*, 2016.

[93] A. Scheerer, T. Schimmer, and T. Kude, "Coordination in large-scale agile software development: A multiteam systems perspective," *47th Hawaii International Conference on System Sciences (HICSS)*, 01 2014.

[94] R. Vallon., K. Bayrhammer., S. Strobl., M. Bernhart., and T. Grechenig., "Identifying critical areas for improvement in agile multi-site co-development," in *Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE,*, INSTICC. SciTePress, 2013, pp. 165–172.

[95] M. Daneva, E. van der Veen, C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar, and R. Wieringa, "Agile requirements prioritization in large-scale outsourced system projects: An empirical study," *Journal of Systems and Software*, vol. 86, no. 5, pp. 1333–1353, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121212003536

[96] P. Ralph and P. Shportun, "Scrum abandonment in distributed teams: A revelatory case," in *Pacific Asia Conference on Information Systems*, 06 2013.

[97] R. Noordeloos, C. Manteli, and H. V. Vliet, "From rup to scrum in global software development: A case study," in *2012 IEEE Seventh International Conference on Global Software Engineering*, 2012, pp. 31–40.

[98] D. Batra, D. VanderMeer, and K. Dutta, "Extending agile principles to larger, dynamic software projects: A theoretical assessment," *J. Database Manage.*, vol. 22, no. 4, p. 73–92, oct 2011. [Online]. Available: https://doi.org/10.4018/jdm.2011100104

[99] J. Cho, "Distributed scrum for large-scale and mission-critical projects." in *Americas Conference on Information Systems (AMCIS)*, vol. 1, 01 2007, p. 235.

[100] M. Korkala and P. Abrahamsson, "Communication in distributed agile development: A case study," in *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)*, 2007, pp. 203–210.

[101] B. Linders, "Don't copy the spotify model," https://www.infoq.com/news/2016/10/no-spotify-model/, 2016, [Online; accessed 19-June-2022].

[102] R. Camara, M. Marinho, S. Sampaio, I. Honório, and H. Moura, "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, pp. 218–232. [Online]. Available: https://sol.sbc.org.br/index.php/cibse/article/view/20974

[103] E. Wenger, R. McDermott, and W. Snyder, *Cultivating Communities of Practice: A Guide to Managing Knowledge*, ser. NetLibrary Inc. Harvard Business School Press, 2002. [Online]. Available: https://books.google.com.br/books?id=m1xZuNq9RygC

[104] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *IEEE Software*, vol. 33, no. 3, pp. 94–100, 2016.

[105] K. Schwaber and J. Sutherland, "The scrum guide: The definitive guide to scrum: The rules of the game," 2020.

[106] K. Beck and E. Gamma, *Extreme Programming Explained: Embrace Change*, ser. An Alan R. Apt Book Series. Addison-Wesley, 2000. [Online]. Available: https://books.google.com.br/books?id=G8EL4H4vf7UC

[107] C. Larman and B. Vodde, "Scaling lean & agile development: Thinking and organizational tools for large-scale scrum," 2008.

[108] R. Streeton, M. Cooke, and J. Campbell, "Researching the researchers: Using a snowballing technique," *Nurse researcher*, vol. 12, pp. 35–46, 02 2004.