UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Rafael da Camara Figueredo

Scaling agile methods in global software projects: Is it possible with SAFe?

Recife

2020

Rafael da Camara Figueredo

Scaling agile methods in global software projects: Is it possible with SAFe?

Monograph presented to the Bachelor of Computer Science Course of the Federal Rural University of Pernambuco as a partial requirement to obtain the Bachelor of Computer Scientist title.

Federal Rural University of Pernambuco – UFRPE

Computing Department

Bachelor of Computer Science Course

Orientador: Marcelo Luiz Monteiro Marinho

Recife

2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

C172s Figueredo, Rafael da Camara
Scaling agile methods in global software projects: Is it possible with SAFe? / Rafael da Camara Figueredo. - 2020.
79 f. : il.

Orientador: Marcelo Luiz Monteiro Marinho.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2020.

1. Agile Software Development. 2. Global Software Development. 3. Agile GlobalSoftware Development. 4. Software engineering. 5. Scaled Agile Framework . I. Marinho, Marcelo Luiz Monteiro, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por Rafael da Camara Figueredo às 14 horas do dia 03 de setembro de 2020, no link meet.google.com/qhb-eytf-mqi, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado **Scaling agile methods in global software projects: Is it possible with SAFe?**, orientado por Marcelo Luiz Monteiro Marinho e aprovado pela seguinte banca examinadora:

Marcelo Luiz Monteiro Marinho

DC/UFRPE

Suzana Cândido de Barros Sampaio
DC/UFRPE

To my family, my girlfriend, and my professors that supported me in this entire journey …

Acknowledgments

First of all, I would like to thank all my family that believed in me since the beginning, which supported and encourage me in everything I have been doing. I want to thank mainly my mother, brother, brother in law and girlfriend, Conceição, Thiago, Bruno, and Larissa to give me all the support and help I had needed before and during the course. Also, I would like to thank my dad, that passed way in 2009, but that had never forgotten to dedicate all of his life to his children and wife.

I am grateful for my brother who is truly an inspiration for me. He is the person that most supported me, most encourage me to follow my dreams, which indirectly proved to me that we can achieve whatever we want. Also, I want to thank him due to the English review he had made of this research.

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

Also, I would like to thank the most inspiring professors I have had during college, Marcelo and Suzana, that support me through the entire process of this research. Without their support, I would never achieve the results of this project.

To professor Marcelo, I want to thank him for being extremely dedicated and committed to everything he does, especially for this research and others he executed with me. Also, I would like to thank him for being one of my professional and personal inspirations, to help me during the bachelor course, and teach me the best of software engineering.

To professor Suzana, I would like to thank her for being such a human with us, understanding all of our points, encouraging us to achieve the best of ourselves, and giving the best tips for the research written. Also, she is a personal and professional inspiration for me, and I want to thank her for all the support and dedication she gave in this project and to better understand my goals and objectives.

I am also thankful for all the professors of the computing department of the Federal Rural University of Pernambuco that had to contribute to my graduation. Specifically, I would like to thank George Valença which was such important for me during the course who always believed in my potential and encouraged me during challenging projects, and also for being such an inspiration. Also, I would like to thank professor Carlos Julian

for being one of the best companies during graduation and for having one of the best teaching didactics.

To all my friends, the ones I have made during the time at the University, Samantha, Thiago Gomes, Fábio, Felipe, Nycolas, Thiago Bastos, Anderson, Davi, Lucas, Yuri and Elthon and Luizinho. I want to thank all of them for being present with me in the good and bad moments and a lot of disciplines. Survive in the University was much easier surrounded by them.

To Samantha, I would like to thank her for being a true partner for everything inside and outside of the University. Since the first semester of the course in 2015, I am with Samantha facing the hardest obstacles of the University and life. I wanna thank her for all the support she gave me during my bad and good moments in my professional and personal life. She is truly a friend that I will take for life.

To Thiago Gomes, I am grateful for being the most crucial person in my professional life as a software developer. I want to thank him since the support he gave me as a monitor of the programming discipline, he opened my mind for other software technologies and encourage me to study it frequently. Also, I would like to thank him for being a true friend in everything, and an inspiring person that has the persistence to achieve anything that wants.

Also, I would like to thank Fabio for being with me in my first internship experience, for all the things I could learn from him, and all the partnerships we had to better face the University. Also, I want to thank him for all the help he gave me during the discipline projects, and for being one of the persons that most helped me to be a better professional developer.

Withal, I would like to thank, Yuri, and Elthon, that directly helped me in the conduction and execution of this research by searching, reading, and extracting data for the systematic literature review.

Moreover, I would like to thank all the work colleagues from Di2win that supported my graduation and also helped me to achieve it, specifically Saulo, Carlos, Paulo, Felipe, Tarcísio, Mariana, Victor, Thiago, Matheus, Luna, Caio, Jorge.

Finally, I am thankful for every one that somehow helped to accomplish my bachelor course.

“A persistência é o caminho do êxito.”
(Charles Chaplin)

Resumo

A desenvolvimento de software global continua crescendo substancialmente e está rapidamente se tornando uma norma do desenvolvimento de software, sendo fundamentalmente diferente do da engenharia de software local. Embora os métodos ágeis terem sido originalmente construídos para times pequenos, o desenvolvimento ágil de software vem se tornando uma opção atraente para empresas que tentam melhorar sua produtividade em times globais. O uso de práticas ágeis no desenvolvimento de software global tornaram-se a principal opção para executar projetos em ambientes distribuídos devido aos seus benefícios relacionados a uma melhor comunicação, coordenação, produtividade e qualidade do projeto. Entretanto, enquanto empresas continuam crescendo, junto a complexidades e desafios que surgem rapidamente, muitos dos seus projetos vem se tornando globais e de larga escala, o que faz elas buscarem constantemente como adaptar o ágil nesses cenários e consequentemente como escalar o ágil para uma melhor coordenação. A literatura atual não fornece uma imagem coesa de como as práticas ágeis são de fato implementadas em um ambiente distribuído, e também como elas podem ser escaladas em projetos globais de larga escala. Faltam informações sobre como *como* usar e escalar o ágil em cenários distribuídos, *quais* práticas ágeis e escaláveis funcionam nas equipes de desenvolvimento de software global, e por fim, *quem* deve aplicá-las. Com base nessa lacuna da literatura, este estudo tem como objetivo destacar como as práticas ágeis são aplicados no contexto de desenvolvimento de software global, a fim de desenvolver um conjunto de técnicas que possam ser relevantes tanto na pesquisa quanto na prática. Além disso, o estudo visa destacar um conjunto de práticas ágeis usadas pelas equipes de desenvolvimento de software global para escalar o ágil e mapear essas práticas com as práticas do “Scaled Agile Framework” (SAFe). Para responder as duas perguntas de pesquisa, sendo a primeira: “Como as práticas ágeis são adotadas nas equipes de desenvolvimento de software global ágil?” e a segunda: “Quais são as práticas relatadas na literatura de desenvolvimento de software global ágil que adotam práticas do SAFe ao adotar o desenvolvimento ágil escalável?”, foi conduzida uma revisão sistemática da literatura nas áreas de desenvolvimento de software ágil, global e global ágil. Uma síntese das soluções encontradas em setenta e seis estudos forneceu 48 práticas ágeis distintas que empresas podem implementar em ambientes distribuídos globalmente. Além disso, dessas 48 prátias ágeis, foi possível identificar 18 práticas ágeis escaláveis que adotam as práticas do SAFe. Essas práticas implementáveis são úteis para fornecer soluções para gerenciar times globais com agilidade, enquanto as práticas do SAFe vinculadas as práticas ágeis escaláveis fornecem diretrizes para melhor escalar o ágil em projetos globais de larga escala.

Palavras-chave: Desenvolvimento de software ágil, Desenvolvimento de software globa, Desenvolvimento software global ágil, Engenharia de software, Ágil em larga escala, Métodos ágeis, SAFe, Revisão sistemática da literatura.

Abstract

Global Software Development (GSD) continues to grow substantially and it is fast becoming the norm and fundamentally different from local Software Engineering development. Withal, agile software development (ASD) have become an appealing choice for companies attempting to improve their performance although its methods were originally designed for small and individual teams. Despite it, agile practices in Global Software Development (AGSD) has become the main option to execute projects in distributed environments due to its benefits of better communication and coordination, improved productivity, and quality. However, while organizations continue to grow, the complexity and challenges arise fast, many companies are dealing with large-scale global projects and looking for how to adapt agile in those scenarios and scaling agile practices to coordinate them. The current literature does not provide a cohesive picture of how the agile practices are taken into account in the distributed nature, and also how to scale than in large-scale AGSD projects. It lacks data on *how* to use agile and also scale it in GSD settings, *which* agile and scaling agile practices work in Global Software Development (GSD) teams and *who* are supposed to apply them. Based on this literature gap, this study aims to highlight how ASD practices are applied in the context of GSD to develop a set of techniques that can be relevant in both research and practice. Furthermore, it also aims to highlight a set of agile practices that are used by GSD teams to scale agile and map those practices with Scaled Agile Framework (SAFe). To answer both of the research questions, first: “How agile practices are adopted in agile global software development teams?”, second: “Which practices reported in AGSD literature embrace practices from SAFe when adopting scale agile development?”. It was conducted a systematic literature review of the ASD, GSD, and AGSD literature. A synthesis of solutions found in seventy-six studies provided 48 distinct agile practices that organizations can implement in globally distributed settings. Furthermore, from those 48 agile practices, it was possible to identify 18 scaling agile practices embrace SAFe practices. These implementable practices go some way towards providing solutions to manage GSD with agility, while the linked SAFe practices provide guidelines for better scale agile in large-scale global projects.

Keywords: Agile Software Development, Global Software Development, Agile Global Software Development, Software engineering, Large-scale agile, Agile methods, SAFe, Systematic Literature Review.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ASD	Agile Software Development
GSD	Global Software Development
AGSD	Agile Global Software Development
SAFe	Scaled Agile Framework
XP	eXtreme Programming
PO	Product Owner
SLR	Systematic Literature Review

Sumário

1 Introduction

Two strong trends shape modern software engineering. First, companies are strongly required to embark on Global Software Development (GSD) endeavors, among other things to attempt to lower production cost and increase the pool of available, talented individuals. Second, agile software development (ASD) is increasingly applied in (distributed) projects to support a more dynamic products handling (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011).

Agile and global software development (AGSD) is currently a significant trend (VALLON et al., 2017). VersionOne of the 14th annual state of the agile report states that 71% of respondents said their organization practices agile with multiple co-located teams collaborating across geographic boundaries (VersionOne, Inc., 2020). AGSD leads to many challenges, given the inherent nature of both paradigms: ASD and GSD. On the one hand, GSD communication is commonly based on documents, i.e., explicit knowledge that decreases the effect of the four distances of this paradigm (physical, temporal, linguistic, and cultural) (MODI; ABBOTT; COUNSELL, 2017). On the other, the agile manifesto (BECK et al., 2001) states that, in ASD, face-to-face interactions are preferable to following strict communication processes and working software is preferable to comprehensive documentation (LOUS et al., 2018b; ALTAF et al., 2019).

Moreover, software development organizations are looking for support on how to implement agile methods, and how to do it within a global environment. However, it is still unclear how to ensure the use of agile methods while the organization continues to grow and the complexity and challenges arise. Therefore, scaling agile remains a challenge in software development, due to the strong coordination needed among teams (RAZZAK, 2016a); the increase in complexity in globally distributed projects(PAASIVAARA, 2017); and the difficulties that arise with the global distance and the successful scaling of agile practices (DIKERT; PAASIVAARA; LASSENIUS, 2016b).

Large organizations usually have large projects carried out by large distributed development teams, which require agile methods to be scaled. According to Leffingwell (LEFFINGWELL, 2007), agile scaling involves many challenges, which include coordination among multiple agile teams, lack of initial architecture, lack of requirement analysis, beyond all the challenges of regular global projects. Thus, several frameworks for scaling agile software development have been suggested, such as Scaled Agile Framework (SAFe) (Leffingwell, Dean, 2020), Disciplined Agile Delivery (DAD) (AMBLER; LINES, 2012), Large-Scale Scrum (LeSS) (LARMAN; VODDE, 2016), Nexus (BITTNER et al., 2017) and Scrum@Scale (Sutherland, Jeff and Brown, Alex, 2020). The

State of Agile Survey (VersionOne, Inc., 2020) shows that 35% of respondents reported the use of SAFe, 4% DAD, 4% LeSS, and 3% Nexus. However, documented experiences regarding the use of these structures are still scarce, like (i) how they are used; (ii) in what kind of circumstances they are most suitable; (iii) and which the success factors and challenges of their use are.

Schwaber (Schwaber, Ken, 2013), one of the creators of Scrum, criticizes SAFe in several ways for being too downward and inflexible, with the risk of suffocating the teams under detailed routines and practices. In the meantime, benefits from adopting SAFe to scale agile (PAASIVAARA, 2017) have been reported. SAFe is becoming popular, being used at several companies which includes BMC Software, Mitchell International, Intel, Hewlett-Packard Enterprise, Cisco, Trade Station Technologies, Discount Tire, John Deere, Valpak, Infogain, SEI (Leffingwell, Dean, 2020), among others.

Use and scale agile is particularly challenging in the global software development context (PAASIVAARA; LASSENIUS, 2011). Although, the authors claim that SAFe is used by large organizations that embrace global software development (Leffingwell, Dean, 2020), more empirical research is needed to evaluate whether SAFe practices are in line with agile practices in global software development (AGSD) literature (VALLON et al., 2017; DIKERT; PAASIVAARA; LASSENIUS, 2016b).

The challenging scenario faced by GSD projects from the use, adaptation, and scaling of agile practices in global software development needs more research. Based on it, this project aims to answer two research questions. First: how are agile practices adopted in agile global software development teams? From those agile practices, we can look for the second question: which practices reported in AGSD literature embrace practices from SAFe when adopting scale agile development? In order to answer both questions, this project consist of a Systematic Literature Review (SLR) (KITCHENHAM; CHARTERS, 2007) that focused on finding research approaches from 2001 to 2019 that highlighted the adaptation and scaling of agile across the globe. A total of 76 research studies were considered for this SLR, from those 76 we could select 19 related to the agile scale. Also, we have presented 48 agile practices identified through SLR, and from those 48 practices, we could link 18 scaling agile practices that address how SAFe can be used to scale AGSD.

This paper contributes to a collection of experiences reported in the literature on how practices were applied to the different ASD in GSD projects to mitigate challenges with distributed software engineering. Besides, this research contributes to the AGSD by presenting a set of scaling agile practices that are used by global teams that embrace SAFe practices. Both results originate from a systematic synthesis of recommendations found in the related GSD.

1.1 Goals

1.1.1 General Goal

Present a set of agile practices used and adapted by agile global software development that are reported in the literature. Also, present the scaling agile practices used by those teams that embrace SAFe and the circumstances they are most suitable.

1.1.2 Specifics Goals

1. Develop a systematic literature review in agile global software development teams to discover how they apply agile practices in globally distributed environments;
2. List the most used agile practices and describe how the practitioners applied it, and who are responsible for each one;
3. Develop the mapping of the AGSD scaling agile practices used by the practitioners that embrace SAFe practices, and also describe how to apply it and who are responsible for each one.

1.2 Document Structure

Beyond this chapter 1, the document has four more chapters that compose this research.

Chapter 2 presents the main subjects regarding this study, and also the related works.

Chapter 3 shows the methodology used in this study, its stages and describes each step executed during the systematic literature review and the mapping.

Chapter 4 presents the results gathered in the research. First, this chapter presents the agile practices extracted from the systematic literature review. Then, the scaling agile practices that embrace SAFe practices are present.

Finally, chapter 5 presents the conclusion and main contributions of this work, the limitations and threats of validity, and the possible future works.

2 Background

This section aims to present a brief theoretical foundation about the subjects related to this project. Also, It intends to give to the reader an understanding of the research areas.

2.1 Agile Software Development - ASD

Agile methods for software development were introduced around the beginning of the new century. The increasing business need for fast creation of the internet and mobile applications was a key driver for the introduction of these lightweight and nimble development processes (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). The agile ideas, promised that higher customer satisfaction can be achieved by addressing such uncertainty aspects and delivering working software frequently with shorter timescale (WAHYUDIN et al., 2008). The ideas of agile software development have gained acceptance in the mainstream software development community. Surveys pointed out that agile teams are often more successful than traditional ones. According to the agile manifesto (BECK et al., 2001), ASD emphasizes individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.

The most widely used methodologies based on agile principals are Scrum and eXtreme Programming (XP) (Jalali; Wohlin, 2010). However, other methods such as Dynamic Systems Development Method, Adaptive Software Development, and the Crystal Family stress upon short time goal and incremental delivery, dividing the entire projects into sprints and every sprint governed by complete software development life cycle (Sriram; Mathew, 2012; HOLE; MOE, 2008). The success of ASD depends significantly on team interaction (DORAIRAJ; NOBLE; MALIK, 2012). Agile methods have enabled software project teams to meet the challenges of an ever turbulent business environment through enhanced flexibility and to emergent customer needs (MARUPING, 2010).

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

2.2 Global software development (GSD)

Global software development (GSD) is the name given to designate companies that distribute their software development beyond their national boundaries (Marinho; Noll; Beecham, 2018). Most of these companies use GSD on a large scale intending to achieve expenses cuts, economic advantages, access to global talent, faster delivery, and a 24-hour software development cycle which is possible due to different time zones (RIZVI; BAGHERI; GASEVIC, 2015). However, while the adoption of GSD has increased in companies, having people in different locations can lead to many problems related to communication, coordination, and control of the development process (Hossain; Babar; Paik, 2009). Furthermore, several studies showed that 40 percent of global software development projects have failed (Betz; Makio; Stephan, 2007) because of those problems.

The major difference between global teams and co-located teams is the fact that team members are not physically present in the same environment at the same time. Although, both teams have the same goals that are to develop and deliver constant value to a project or product. However, the fact that team members are spread in other countries leads to a lot of more complications than co-located teams faces, such as asynchronous communication, lack of face-to-face communication, difficult in constructing a shared vision with the team, difficult in arranging meetings due to different time zones, and different knowledge levels about a common language (Marinho; Noll; Beecham, 2018).

2.3 Agile Global software development (AGSD)

The presence of agile practices in AGSD was seen since 2002 (JALALI; WOHLIN, 2012), one year after the launch of the Agile Manifesto (BECK et al., 2001). Furthermore, the use of agile practices is also related to the success of GSD projects (RAMESH et al., 2006; RAZZAK et al., 2017; Hossain; Babar; Paik, 2009).

However, achieving success in a GSD project is a difficult task, which can only be achieved through planning, organization, effective personnel, leadership, control, coordination, and project management. These practices aim to mitigate the geographic, temporal, and sociocultural distances (PAASIVAARA; LASSENIUS, 2010). To overcome most of the complications faced by teams distributed globally, companies are adopting agile methods for their distributed environment. However, by definition, agile practices have been developed to help co-located teams (VALLON et al., 2017). Although many studies demonstrate that agile methods can be useful to mitigate GSD problems (RAMESH et al., 2006; RAZZAK et al., 2017; Hossain; Babar; Paik, 2009). In this perspective, agile methods appear as a viable methodology, since it is increasingly used in

software development worldwide.

The practices adopted in distributed teams are adapted (VALLON et al., 2017) so that they can deal with the challenges of the GSD. Face-to-face meetings and pair programming, for example, are not possible to be implemented the way they are defined, as members are in different locations, as well as different time zones (RAJPAL, 2018). In this way, it is up to the teams to carefully select different asynchronous and synchronous communication tools for the implementation of these practices. The adaptation and selection minimize the challenges caused by the GSD, as well as promoting control and coordination in distributed teams (Szabó; Steghöfer, 2019). Among the agile approaches, in the context of AGSD, the most used framework is Scrum followed by Kanban (MARINHO et al., 2019).

2.4 Scaled Agile Framework®

In the road to ease the insertion of agile in globally distributed teams many companies choose frameworks to scale agile practices, such as the Scaled Agile Framework (Leffingwell, Dean, 2020).

The SAFe (Leffingwell, Dean, 2020) was developed by Dean Leffingwell in 2012, and it focuses on scale agile on large enterprises. The framework is a documented and proven approach to scale agile practices, strategies and benefits in large enterprise environments. It was developed to help companies to manage, control and organize the development process in a context with many teams and many people (RAZZAK et al., 2018; Paasivaara, 2017).

SAFe framework is a solid structure that covers all organization levels. Its four core values alignment, built-in quality, transparency, and program execution (Leffingwell, Dean, 2020) are associated with values and principles from Scrum (SCHWABER; BEEDLE, 2001), eXtreme programming (BECK, 2000), Lean Software Development (POPPENDIECK; POPPENDIECK, 2003) and the Agile Manifesto (BECK et al., 2001). Recently, SAFe 5.0 was launched (Leffingwell, Dean, 2020). This new release aims to cover all the enterprise levels and enable business agility. Furthermore, it combines program and team level to form the Essential level, and brings focus on customer centricity, design thinking, and the continuous delivery pipeline.

The SAFe framework is now structured in three layers:

1. **Portfolio**: This is responsible for the funding and the strategic analysis of the diverse initiatives present in the enterprise (Leffingwell, Dean, 2020).
2. **Large Solution**: It describes the roles, practices and guidance for enterprises that build large and complex solutions, like critical systems (Leffingwell, Dean, 2020).

3. **Essential**: this combines the past layers (program and team). It provides a starting point for SAFe implementation and contains the practices, roles, events, and artifacts necessary to deliver business solutions through ART that is a team of agile teams (Leffingwell, Dean, 2020).

2.5 Related Work

One of the first secondary studies about the use of agile methods and practices in GSD was conducted by Jalali and Wohlin (JALALI; WOHLIN, 2012). In their article, we could saw that agile became more popular in a distributed environment from 2004 and forward. Also, from 63 studies selected, 53 reported success in their distributed projects using agile practices. Besides, Jalali and Wohlin identified 25 agile practices in distributed environments, although they not present how to apply those practices and only considered the practices used in successful cases and not on failure reports. Moreover, this study was conducted 10 years ago and compiled papers until 2009. Since one decade has passed from this study, an update will be well received. Finally, the authors reported that there was insufficient evidence, at the time, about scaling agile in large distributed projects, which reinforces the need to research the present scenario.

Vallon *et al.* (VALLON et al., 2017) continued Jalali and Wohlin study (JALALI; WOHLIN, 2012) following a similar study design, to analyze and compare the newer results with the past ones. The study aimed to synthesize the state of the art of successful agile practices applied in GSD. The main contribution of this study was to update the state of the art of agile practices in GSD covering publication from 2009 to 2016. The authors selected 145 studies, the authors selected 145 studies, but only 89 were read due to be successful cases of agile applicability in GSD. It was reported that Scrum is the most used agile framework in a distributed environment. Besides, the study presents 21 practices reported in AGSD projects, and compare the frequency of each practice to Jalali and Wohlin SLR study. However, similar to them, agile practices in GSD were not described in a way of how to apply it in GSD, and also only successful cases were present. Indeed, the study presents an update of the state of the art about agile practices in GSD projects. But, it still lets a gap opened, the evidence of scaling agile in large distributed projects were present, although a description of scaling agile practices was not reported. Finally, Vallon *et al.* (VALLON et al., 2017) reinforces the need to present empirical evidence on how to apply agile practices in GSD and how to scale it.

Hossain *et al.* (Hossain; Babar; Paik, 2009) conducted a SLR on Scrum usage in GSD. The study summarizes a set of challenges that arise from Scrum in GSD and strategies to deal with them. It was only researched scrum practices related to the project management area in primary studies from 2003 to 2009. The practices

were categorized and information about how to apply them in distributed settings was also reported. However, the authors only considered GSD studies that applied Scrum, which may have caused a loss of information regarding agile practices of other agile methods/frameworks in GSD projects. Besides it, from 2009 to now the research area had received huge contributions from practitioners and researchers that could present new and updated results. Finally, the study does not present ways to scale scrum practices in AGSD projects.

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

Focused on challenges in distributed settings, Alsahli *et al.* (ALSAHLI; KHAN; ALYAHYA, 2017) presented an SLR study with a focus on GSD challenges and agile practices based on Scrum used in GSD to mitigate those challenges. Studies from 2006 to 2016 were selected, and the authors reported how the GSD projects implemented agile practices in GSD, to mitigate GSD challenges. However, in the hole process only agile practices related to the mitigation of GSD challenges stated by the papers selected were considered. Such a decision reduced the number of papers selected for 24, and the emphasis given to Scrum reduced the set of agile practices applied in GSD settings. Besides, the study discusses the use of agile practice in large-distributed projects but does not go deep into scaling agile practices.

Another SLR study related to AGSD challenges, specifically for communication challenges, was conducted by Alzoubi *et al.* (ALZOUBI; GILL; AL-ANI, 2015). The study aimed to extract the communication challenges faced by AGSD teams and techniques to deal with them. The study considered papers from 2006 to 2014, the communication challenges were classified into six categories and both impact and techniques to face them were present and described. Besides, scale agile were considered to deal with distance issues. However, different from our study Alzoubi *et al.* (ALZOUBI; GILL; AL-ANI, 2015) does not take into account agile practices in GSD settings and how to apply

them.

According to Razavi and Ahmad (Razavi; Ahmad, 2014), the practitioners are concerned about scale agile since 2005, and the top topics about large and distributed organizations of software development are scaling up agile development, transition to large scale, agile implementation in a large organization, etc. However, Razavi and Ahmad (Razavi; Ahmad, 2014) just describes the general topics about large and distributed organizations. It presents the needs and current gaps of the research topic, although the study does not discuss how to adapt and scale agile practices in those organizations.

With more focus on large-scale agile development, Dikert *et al.* (DIKERT; PAASIVAARA; LASSENIUS, 2016a) conducted an SLR to review how large-scale agile transformations occurred in the industry and also describes how agile methods and lean software development were adopted at scale. The study focuses on identify challenges and success factors during the agile transformation of large-scale and distributed scenarios. The article points out that 90% of the selected studies for the SLR were experience reports, which indicates that academic research is necessary on the topic. However, since the focus was mostly given to the challenges and success factors of scaling agile, specific agile practices and how to implement it in GSD settings were not taken into account. Additionally, the paper does not relate the agile practices covered in success factors to the SAFe practices. Although it points to the need to better understand how scaled frameworks are used, what are the challenges and benefits from them, and how their practices are tailored. Such points indicate the need for further research in large-scale agile distributed environments topic.

Based on the need for further research on large-scale agile at distributed settings. Razzak *et .al* (RAZZAK et al., 2017) conducted a study to evaluate the adoption rate of SAFe practices in a small to medium sized enterprises (SME). In the study, the authors conducted surveys to better understand how SAFe can be implemented in a global SME. Besides, the study gives focuses on the health evaluation of five areas from the SAFe team level. The survey was used to get answers from team members about product ownership, technical, program increment/release, team, and sprint health. The health of those areas is based on the rate of practices adopted and reported by the members. The study presents valuable information to the area, although it focuses on the comparison of the practices adoption rate between two quarters. Not to mention, a proper description of how the SAFe practices were applied is not taken into account.

Finally, it is possible to conclude that the knowledge about what agile practices are being adopted by GSD teams can be truly helpful for practitioners and researchers. Thus, it can improve the development process in distributed settings, since the lastest SLRs' were conducted a few years ago and almost none of them describes how to apply

the agile practices in GSD settings. Furthermore, it is important to point which scaling agile practices are being used by the AGSD projects, and which of them are adopting SAFe practices. It is necessary to combine those practices with the proper descriptions on how to apply agile and scale agile in GSD to lead companies to achieve success in globally distributed projects and also reduce the existing gap on how to adopt agile in large-scale global distributed environments.

2.6 Chapter Conclusion

This chapter contextualizes the reader by presenting the main research areas of this study. Also, the section aims to provide a better understanding of the main subjects.

The contextualization sections initiate with a brief overview of agile software development by presenting when it was launched and its main applicability. Further, the concept of global software development was present together with its benefits and main challenges.

Moreover, the combination of agile and GSD which generates AGSD was discussed. AGSD represents the use and adaptation of agile practices in globally distributed environments. In this section, the agile practices are presented as viable solutions to mitigate the common GSD issues. Although, since agile practices were developed for co-located teams, it is necessary to adapt them to AGSD contexts. Also, the SAFe framework was briefly described, showing the three layers that form the new SAFe 5.0, and its four core values.

Finally, the reason for this study was discussed by showing the existing gaps in the literature and the need for an updated SLR on the research topic. Also, it was seen the need for proper description of what agile practices are being used in GSD settings and how to apply and scale them in distributed settings and also in large-scale distributed environments.

3 Methodology

This chapter describes the research methodology chosen for this study and presents the two stages that this study was conducted through. In the first stage, the authors executed a systematic literature review to gather agile and scaling agile practices that were being applied in GSD projects. Later, the found practices were described, and in the second stage, the authors linked the scaling agile practices from the reviewed studies to the practices present in SAFe (Leffingwell, Dean, 2020).

For a better understanding of the research steps of this work, figure 1 is going to show its process and all the phases of this research.

3.1 Systematic Literature Review

In this study, a systematic review approach was adopted to examine the relevant literature. Our goal was to select a sufficient collection of studies to enable the identification of recurring themes.

Established systematic review guidelines (KITCHENHAM; CHARTERS, 2007) recommend that a reviewer carry out the following steps: (i) identify the need for a systematic literature review; (ii) formulate review research question(s); (iii) carry out a search for relevant studies. (iv) assess and record the quality of included studies; (v) classify the data needed to answer the research question(s); (vi) extract data from each included study; (vii) summarise and synthesize study results (meta-analysis); (viii) interpret results to determine their applicability; (ix) write up the study results as a report.

The SLR sought to answer the following research questions: *“How are agile practices adopted in agile global software development teams?”* and *“Which practices reported in AGSD literature embrace practices from SAFe when adopting scale agile development?”*. The Boolean search string used to ensure that a wide variety of papers would be captured is present in table 1:

Main search string
(“Agile” OR “scrum” OR “extreme programming” OR “pair programming” OR “hybrid” OR “lean development” OR “lean software development” OR “SAFe” OR “Scaled Agile Framework”) AND (“global software engineering” OR “global software development” OR “distributed software engineering” OR “distributed software development” OR “GSE” OR “GSD” OR “distributed team” OR “global team” OR “dispersed team” OR “spread team” OR “virtual team” OR “offshore” OR “outsource”)

Tabela 1 – Research string.

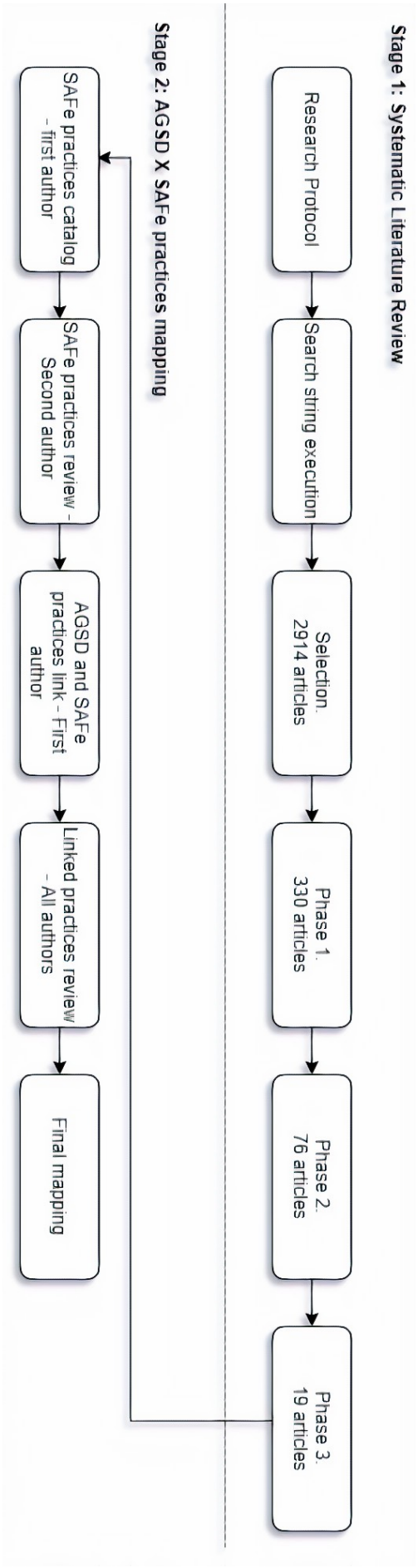

Figura 1 – Research method diagram.

We used this string to search the metadata relating to journals and conference proceedings in IEEEXplore, ACM Digital Library, SpringerLink, Scopus, and Wiley bibliographic databases.

The SLR was conducted by four researchers, myself, my advisor, and two other students from the research group, FREVO. During the SLR's phases, each researcher had a specific function, the research protocol was developed by all researchers and the search string was executed in the bibliography databases by myself and the advisor. The search results were exported as BibTeX files, then it was organized into the StArt software (ZAMBONI et al., 2010), an open-source support tool for SLR research, and then all the articles were evaluated until the full read phase.

Into the selection phase, all the researchers, except the advisor, read the title and abstract of all papers, and then the result dataset of each researcher was discussed. Whether an article was approved by two or all the researchers it would be included for the next phase, although, if a paper received only one vote for approval, such paper would be discussed by all the researchers until they reach a consensus. At phase 1, the 330 papers were split into three groups, and each researcher, except the advisor, was responsible for each group. During this moment each paper was downloaded and evaluated by the researchers, and after the evaluation, the selected ones were evaluated following the same rules of the selection phase.

In phase 2, the 76 articles were split into three groups again, and each researcher, except the advisor, read the full papers of their group. The data were extracted by the researchers in the form of quotes, the dataset of quotes was evaluated by all the researchers and any disagreements were discussed until a consensus reached. Finally, the third phase was conducted by myself who extracted 19 articles from the 76 papers from the past phase that were related to large-scale agile development projects. The duration of each phase is present in Table 2.

Phases	**Period**
Research Protocol	September, 2019
String Execution	October, 2019
Selection Phase	October, 2019 to January, 2020
Phase 1	February, 2020 to March, 2020
Phase 2	April, 2020 to May, 2020
Phase 3	June, 2020 to July, 2020

Tabela 2 – Research periods.

3.1.1 Document selection

The search produced 2914 references (IEEE = 737; ACM = 155; Springer= 322; Scopus = 847; Wiley = 853; Duplicates = 337). The idealized selection process had two components: (phase 1) an initial selection of research results that could reasonably satisfy the selection criteria (outlined next) based on a reading of the articles' titles and abstracts; followed by (phase 2) a final selection against these criteria from the initially selected list of papers based on a reading of their introductions and conclusions.

3.1.1.1 Inclusion/Exclusion criteria

The following criteria guided the selection of papers that helped us address the research questions.

We *included*: (i) complete, peer-reviewed, published articles; (ii) Papers directly related to the research questions; (iii) Papers addresses agile practices in GSD and that the study is available via the university library services accessible to the authors during the time of the search.

We *excluded*: (i) texts not published in English; (ii) Technical content, without proven scientific relevance, eg: editorials, tutorials, key-note speech, white papers, thesis, dissertations, technical reports, books; (iii) Short papers (<=4 pages); and (iv) articles that are not clearly related to the research questions.

Before accepting a paper into the final set for review, we checked to ensure that there was no replication. For example, if a given study was published in two different journals with a different order of primary authors, only one study would be included in the review; this would usually be the most comprehensive or recent study. Besides, we checked to ensure that there was no duplication. For example, in the same paper listed in more than one database, only one study would be included in the review.

With these criteria, we identified 337 duplicate articles and no replicates. After excluding duplicate results from the dataset, we identified articles for inclusion in the initial selection (phase 1). Out of these articles, 330 were passed on to phase 2, in which 252 were eliminated and 76 were finally passed on to the data extraction and data synthesis phase (See Table 3).

The following table shows the number of studies extracted from each engine through the search string, and how many were accepted in each phase of the extractions.

Engine	Selection	Phase 1	Phase 2
ACM	155	42	21
Scopus	847	135	12
Wiley	853	9	0
Springer	322	30	24
IEEE	737	114	19
Total	**2914**	**330**	**76**

Tabela 3 – Papers by engine.

Those 76 articles helped to find the necessary data to answer both research questions, although mainly the first one. However, the final set of articles was also classified in a third phase to identify which papers were related to scaling agile (See table 4).

Engine	Selection	Phase 1	Phase 2	Phase 3
ACM	155	42	21	3
Scopus	847	135	12	1
Wiley	853	9	0	0
Springer	322	30	24	10
IEEE	737	114	19	5
Total	**2914**	**330**	**76**	**19**

Tabela 4 – Papers by engine phase 3.

3.1.2 Study Quality

The quality assessment criteria adopted for our study are based on principles and good practices established for driving empirical research in software engineering (DYBA; DINGSOYR; HANSSEN, 2007), are briefly summarised as follows. We answered the following questions using *Yes, No, Partially*: (i) Is there a clear definition of the study objectives?; (ii) Is there a clear definition of the justifications of the study?; (iii) Is there a theoretical background about the topics of the study? (iv) Is there a clear definition of the research question (RQ) and/or the hypothesis of the study?; (v) Is there an adequate description of the context in which the research was carried out?; (vi) Are used and described appropriate 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) Is provided by the study clearly answer or justification about RQ/hypothesis?; (x) Is provided by the study clearly stated findings with credible results? (xi) Is provided by the study justified conclusions?; and (xii) Is provided by the study discussion about validity threats?

3.1.3 Data Extraction

We examined each selected publication to extract the following elements: (i) study aim or research question, (ii) other results relevant to the study, and (iii) potential themes emerging from the study's conclusions.

We synthesized the data by first identifying each paper's agile practices as to how to apply it to the GSD team. As we gave each occurrence the same weight, the frequencies presented simply reflect how many papers mention a given practice; frequencies, therefore, reflect the prevalence of a theme and not its potential importance.

We classified all included papers into one of the following research types facet derived from Wieringa *et al.* (WIERINGA et al., 2006).

- **Opinion:** Do not describe new research result. Those studies contain the author's opinion statements not grounded in empirical data, related work, or research methods;
- **Solution:** Proposal solution to a problem, it can be an improvement of a technique or the proposal of a new technique. The proposed solution must be argued, and when possible, tested, and validated;
- **Philosophical:** Proposes a new way of thinking/looking to things. It can be a new conceptual framework, a taxonomy, and a secondary study, us has RSL or SMS;
- **Evaluation:** Evaluation of a problem in practice, or evaluations of a technique implemented in practice, e.g case studies;
- **Experience:** It describes personal experiences. It can be personal experiences from the authors, or industrial experience reports.

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

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

The data extracted using the web form was copied to a spreadsheet for data synthesis. Additionally, data synthesis is divided into quantitative and qualitative synthesis.

3.2 Mapping

The first phase for the mapping of scaling AGSD practices and SAFe practices was to access the SAFe website home page (*https://www.scaledagileframework.com/*), which contains the big picture of the framework and a diagram map of the structure. Then follow each reference link from the big picture of SAFe that leads to other pages that describe the element which can be a highlight, a role, an event, or an artifact. Accessing each reference link it was possible to read it, and then identify and extract practices from SAFe. After the extraction of those practices, it has been organized into a hierarchy mirroring the organization of papers.

Since a comprehensive catalog of SAFe approach practices was build, the second phase consisted of a review made by the advisor in the content, and after it, some adjustments were made. By the end of the content review, the catalog of SAFe practice was finished and the third phase started by each SAFe practice being compared to the set of AGSD practices extracted during the systematic literature review. Whether a SAFe practice could be considered to contribute to the implementation of an AGSD practice, then that SAFe practice was linked to it. By the time the practices were linked, the fourth phase consisted of a last review by the advisor, and the last adjustments pointed by him were made.

Finally, through the mapping, we could get a comparison among scaling AGSD practices and SAFe practices, to combine them and define which scaling agile practices used by AGSD projects embrace SAFe practices.

3.3 Chapter conclusion

This chapter describes the research methods used for the execution of this research. The research methodology consisted of two stages (See figure 1), each stage with their own research question. Each stage was described with what was done.

4 Research Development

This chapter presents the results of systematic literature reviews. It includes the extracted agile practices used ing GSD, then the scaling agile practices used by GSD teams linked with SAFe practices (Leffingwell, Dean, 2020).

Primarily, we will discuss and describe the 48 agile global software development practices extracted from the 76 studies from the period of January 2001 to December 2019 that answer the question “how are agile practices adopted in agile global software development teams?”. The overview of the primary studies is present in section 4.1 and qualitative analysis in section 4.2.

Moreover, we describe the 18 scaling AGSD practices linked with many SAFe (Leffingwell, Dean, 2020) practices that answer the question “Which practices reported in AGSD literature embrace practices from SAFe when adopting scale agile development?”.

4.1 Overview of the primary studies

The distribution of research methods types is shown in Figure 2. It can be seen that The majority of the papers used a qualitative approach in their studies (54 papers), followed by a mixed approach (18 papers), and a quantitative method (3 papers). We could not define the type of research method of only one paper, because of it, we got an unclear one.

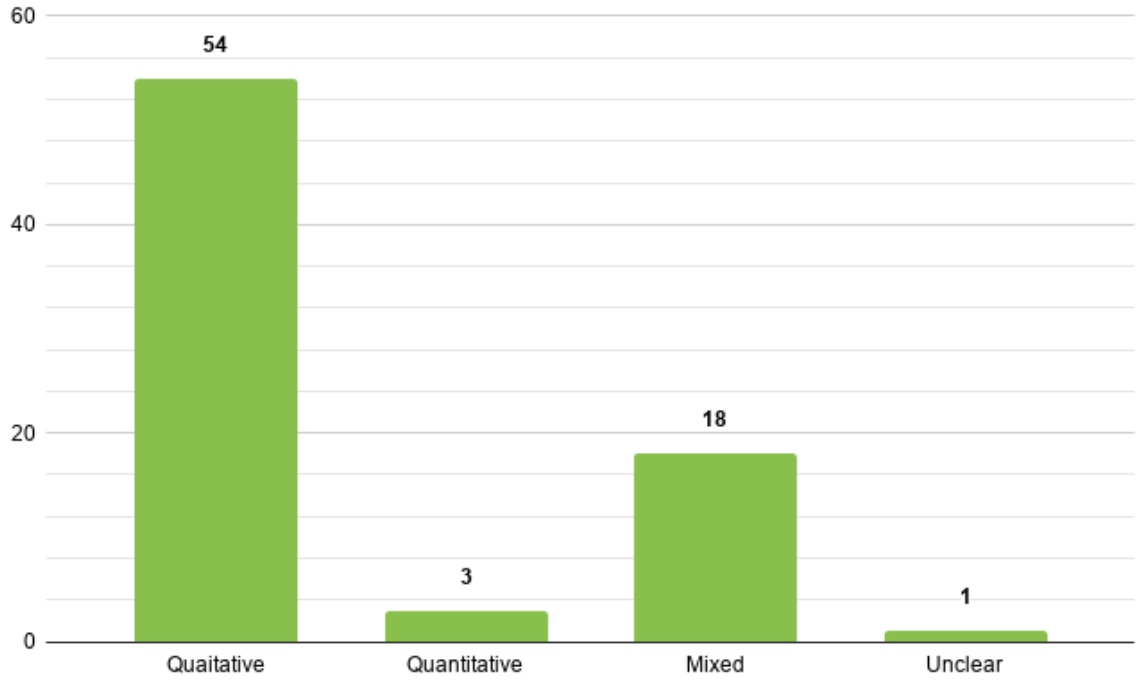

Figura 2 – Research methods types distribution.

The mapping of paper types in Figure 3 shows the research type facets of all studies from the SLR throughout 2004-2019. The research type facets are derived Wieringa *et al.* (WIERINGA et al., 2006), and it aims to classify the papers into classes regarding the research types. It can be seen that at the beginning of the 2000s no studies were found about agile practices in GSD. However, from 2004 to 2011, the number of studies started to emerge and rise, and most of then were from experiences, evaluations, and solutions. Later, in the period of 2012-2019, we can see the appearance of more studies, specifically philosophical studies (13 papers) that indicate a certain maturity of the research field. Although, experience papers (17 studies) continued to be reported what shows that the research field continues to receive attention from the academy who are regularly researching the area. Furthermore, the most common publication types throughout 2004-2019 were evaluation papers(25 papers), experience reports (24 papers), and philosophical studies (16 papers).

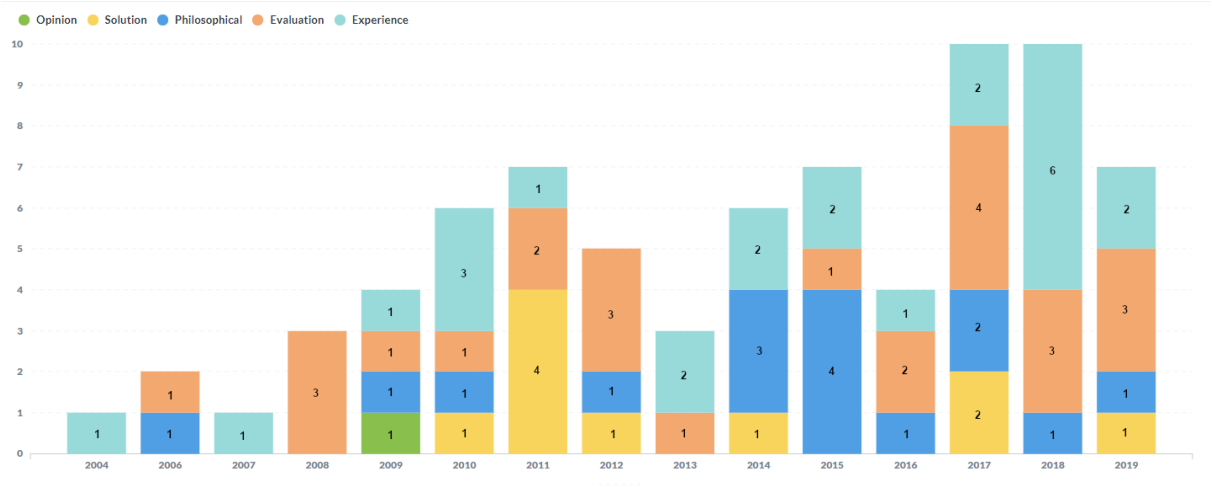

Figura 3 – Research type facets over time.

The distribution of contribution type facets of the reviewed studies derived from Petersen *et al.* (PETERSEN et al., 2008) and is presented in Figure 4. Those facets classifies the studies regarding their contribution to the literature. As we can see the most common contribution types were guidelines (28 papers), then lessons learned (24 papers), followed by advice (12 papers), frameworks (8) papers, and model (4 papers).

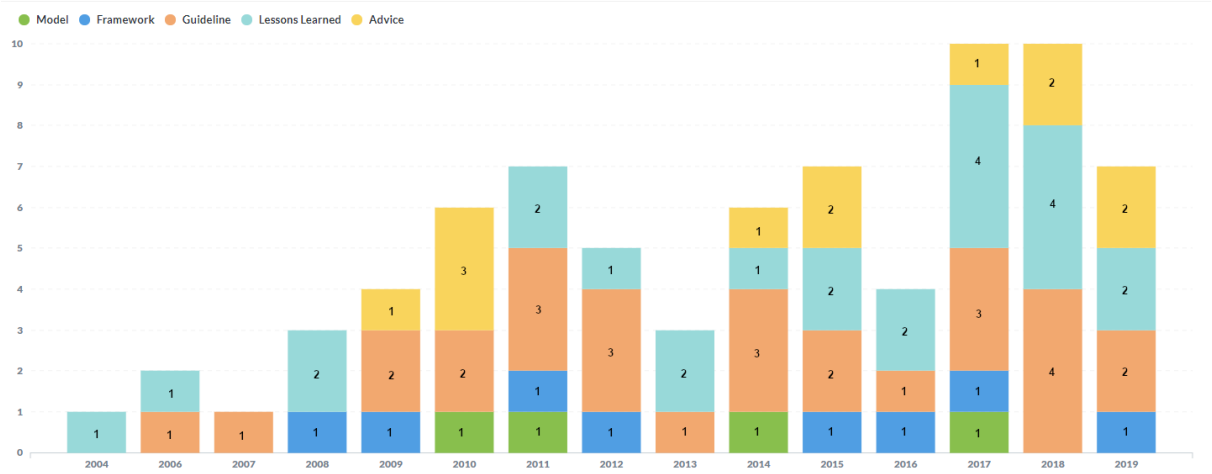

Figura 4 – Contribution type facets over time.

The distribution of methods per year is shown in figure 5. Beforehand, there were articles that used more than one methodology, so the number of methods does not correspond to the number of articles. It can be seen that most articles used the case study methodology (42 articles), where at least 1 article was found, with the exception of the year 2016, followed by interviews (26 articles). The least used methodologies were multiple-Case Study and action Research, with 1 article each in 2010 and 2016 respectively.

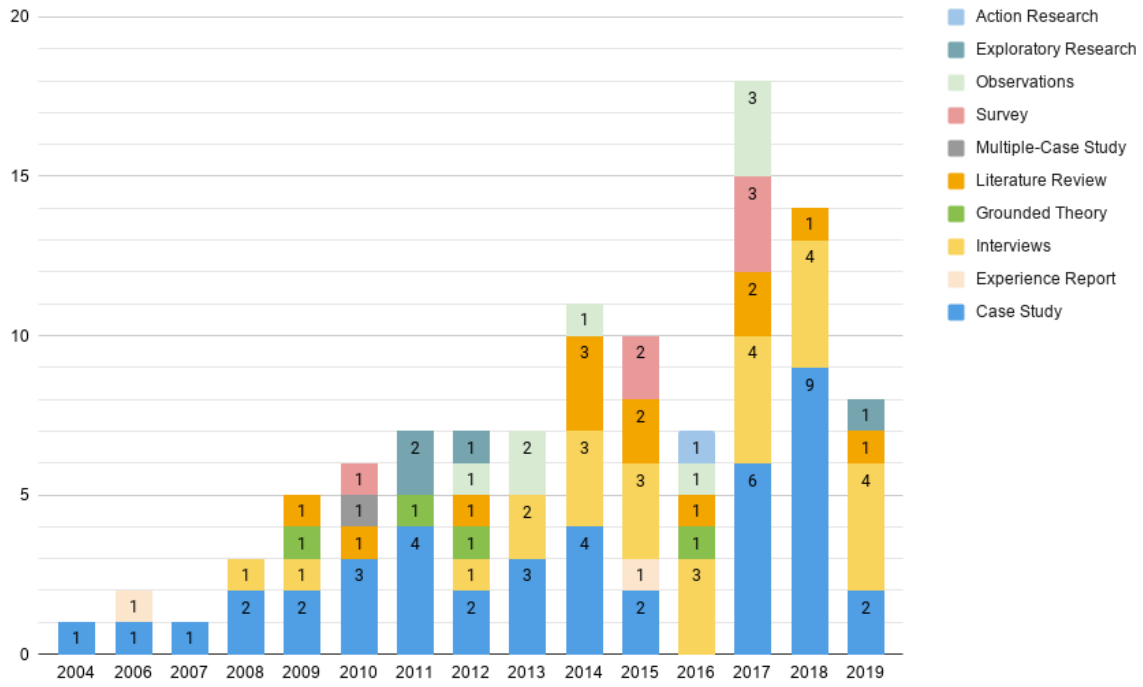

Figura 5 – Distribution of research methods by year.

Each study was assessed independently, according to seven possible quality criteria (see Section 3.1.2). The papers were evaluated on the following scales: <20%, poor; 20%-40%, fair; 40%-60%, average; 60%-80%, good; and >80%, excellent; these are listed in Table 5.

	Poor (<20%)	Fair (20%-40%)	Average (40%-60%)	Good (60%-80%)	Excellent (>80%)
Number of Studies	0	1	5	13	57
Percentage of Papers	0	1,3%	6,5%	17,1%	75%

Tabela 5 – Quality assessment.

4.2 RQ: How are agile practices adopted in agile global software development teams?

To help implement our set of identified practices, we will describe each practice, in the next subsections, based on the review papers. The description aims at the goal of the practice, who should be applying the practice, and how the practice can be adopted in a GSD environment.

4.2.1 Daily meeting (36)

Daily meetings are an excellent way for distributed projects to disseminate information about the status of the project across team members. In a distributed environment is usually held through videoconferences, telephones, etc (Robinson, 2019; Khmelevsky; Li; Madnick, 2017; HUBER; DIBBERN, 2014). *Goal*: it aims to disseminate knowledge in the team about what was done in the previous day, identify impediments, and inform the work to be carried out on the day that begins. *Who*: teams and scrum master. *How*: when it comes to GSD, the studies mentioned that the meetings were held by telephone, video conference, webcam, emails, internet chats among other means of communication (Robinson, 2019; RAZZAK et al., 2018; LOUS et al., 2018b; Szabó; Steghöfer, 2019; LOUS; KUHRMANN; TELL, 2017; HOSSAIN; BANNERMAN; JEFFERY, 2011b; LOUS et al., 2018a; S.; KUMAR; MANI, 2018; LAL; CLEAR, 2018; BJØRN; SØDERBERG; KRISHNA, 2019; VALLON et al., 2017; Khmelevsky; Li; Madnick, 2017; KAUSAR; AL-YASIRI, 2017; RAZZAK et al., 2017; MODI; ABBOTT; COUNSELL, 2017; Gupta; Manikreddy, 2015; MOE et al., 2015; RIZVI; BAGHERI; GASEVIC, 2015; ESTáCIO; PRIKLADNICKI, 2014; HUBER; DIBBERN, 2014; Sundararajan; Bhasi; Vijayaraghavan, 2014; VALLON et al., 2013; RALPH; SHPORTUN, 2013; Sriram; Mathew, 2012; DORAIRAJ; NOBLE; MALIK, 2012; HOSSAIN; BANNERMAN; JEFFERY, 2011a; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; PAASIVAARA; LASSENIUS, 2010; HOSSAIN; BABAR; VERNER, 2009a; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; HOLE; MOE, 2008; KUSSMAUL; JACK; SPONSLER, 2004; BASS, 2015; LEE; YONG, 2009; Jalali; Wohlin, 2010; Hossain; Babar; Paik, 2009).

4.2.2 Communication practices (34)

Aim to replace the common face-to-face communication of a co-located agile team, the distributed teams usually use synchronous communication through videoconference, telephones, and web conferences (PAASIVAARA; LASSENIUS, 2010; HOSSAIN; BABAR; VERNER, 2009a; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008). However, asynchronous communication is also very important in globally distributed teams due to each team is in a different timezone. To enhance asynchronous communication in distributed projects the studies suggested the use of email, chat tools, wikis (KAUSAR; AL-YASIRI, 2017; Hamid, 2013; HOSSAIN; BANNERMAN; JEFFERY, 2011a). *Goal*: it intends to reduce misunderstandings, enhance the bonds of the distributed teams, and share common information among all sites. *Who*: teams. *How*: some GSD projects used the practice of “multiple communication modes” that combines a variety of synchronous and asynchronous approaches to establish a communication strategy in the project(LOUS; KUHRMANN; TELL, 2017; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; HOSSAIN; BABAR; VERNER, 2009b). Furthermore, some papers sug-

gest that distributed teams should synchronize work hours to achieve good communication levels (Robinson, 2019; RAZZAK et al., 2018; Szabó; Steghöfer, 2019; HOSSAIN; BANNERMAN; JEFFERY, 2011b; RAMESH et al., 2006; RICHTER; RAITH; WEBER, 2016; HOSSAIN, 2019; VALLON et al., 2017; KAUSAR; AL-YASIRI, 2017; LAUREN, 2015; BANIJAMALI et al., 2017; Gupta; Manikreddy, 2015; MOE et al., 2015; RIZVI; BAGHERI; GASEVIC, 2015; HUBER; DIBBERN, 2014; Razavi; Ahmad, 2014; Hamid, 2013; DORAIRAJ; NOBLE; MALIK, 2012; DUMITRIU; OPREA; MESNITA, 2011; HOSSAIN; BANNERMAN; JEFFERY, 2011a; PAASIVAARA; LASSENIUS, 2010; MARUPING, 2010; AVRITZER; BRONSARD; MATOS, 2010; HOSSAIN; BABAR; VERNER, 2009a; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; HOLE; MOE, 2008; CHO, 2007; KUSSMAUL; JACK; SPONSLER, 2004; BASS, 2015; Hossain; Babar; Paik, 2009; Nuevo; Piattini; Pino, 2011).

4.2.3 Planning (24)

In this practice, the distributed teams discuss the requirements, tasks, and the necessary work that needs to be done to achieve a desired goal in the project. Planning meetings are mostly conducted frequently to refine and discuss information regarding tasks and customer expectation (Sundararajan; Bhasi; Vijayaraghavan, 2014; VALLON et al., 2013; PAASIVAARA; LASSENIUS, 2010). *Goal*: in the planning meetings, developers have the opportunity to ask questions about proposed tasks, reduce misunderstandings, and align the scope according to customer expectations and feedback. *Who*: dev teams, product owner, scrum master, stakeholders. *How*: global software planning is applied with tools such as Skype, phone calls and videoconferences, e-mails, and visits (Szabó; Steghöfer, 2019; HOSSAIN; BABAR; VERNER, 2009a; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008). Furthermore, according to Rizvi *et al.* (RIZVI; BAGHERI; GASEVIC, 2015), those plans take place in short cycles, to reduce risks and increase feedback from customers and other teams (LOUS; KUHRMANN; TELL, 2017; HOSSAIN; BANNERMAN; JEFFERY, 2011b; LAL; CLEAR, 2018; VALLON et al., 2017; Khmelevsky; Li; Madnick, 2017; KAUSAR; AL-YASIRI, 2017; BANIJAMALI et al., 2017; RAZZAK et al., 2017; MOE et al., 2015; Britto; Mendes; Börstler, 2015; TRIPATHI et al., 2015; BRITTO; USMAN; MENDES, 2014; Sundararajan; Bhasi; Vijayaraghavan, 2014; VALLON et al., 2013; TANNER; CHIGONA, 2012; HOSSAIN; BANNERMAN; JEFFERY, 2011a; PAASIVAARA; LASSENIUS, 2010; AVRITZER; BRONSARD; MATOS, 2010; LEE; YONG, 2009; Jalali; Wohlin, 2010).

4.2.4 Scrum of scrums - SoS (20)

Aims through the scrum master of each team, to hold meetings to share information and keep all members up to date on product events (Gupta; Manikreddy, 2015; VAL-

LON et al., 2013; DUMITRIU; OPREA; MESNITA, 2011). *Goal*: the purpose of the scrum of scrums is to keep teams up to date on the events of the project. *Who*: scrum masters and teams. *How*: in the context of GSD, to apply SoS, teams are formed based on their location, with a representative member (Scrum master) on each team to ensure communication among teams (KAUSAR; AL-YASIRI, 2017; Hossain; Babar; Paik, 2009). According to Passivaara and Lassenius, it was observed that SoS meetings can be similar to the scrum meetings with the need to organize the meetings virtually (PAASIVAARA; LASSENIUS, 2010). Also, at the scrum of scrums meeting, project progress, resource decisions, prioritization, testing approaches, shared coding infrastructure, and people management are discussed (LOUS; KUHRMANN; TELL, 2017; HOSSAIN; BANNERMAN; JEFFERY, 2011b; S.; KUMAR; MANI, 2018; GUPTA; JAIN; SINGH, 2018; VALLON et al., 2017; Khmelevsky; Li; Madnick, 2017; Gupta; Manikreddy, 2015; RIZVI; BAGHERI; GASEVIC, 2015; Sundararajan; Bhasi; Vijayaraghavan, 2014; VALLON et al., 2013; LI; MäDCHE, 2013; DUMITRIU; OPREA; MESNITA, 2011; HOSSAIN; BANNERMAN; JEFFERY, 2011a; AVRITZER; BRONSARD; MATOS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; LEE; YONG, 2009; DINGSøYR et al., 2017; Nuevo; Piattini; Pino, 2011).

4.2.5 Visits among sites (20)

Regular Visits among sites are a practice where team members come together to reduce the sociocultural distance, improve communication, and achieve better Collaboration among teams (DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; LASSENIUS, 2010; Hossain; Babar; Paik, 2009). *Goal*: reduce sociocultural distance, improve communication, and achieve better collaboration among teams *Who*: teams and stakeholders. *How*: when affordable, the visits among sites are dedicated to face-to-face meetings (Gupta; Manikreddy, 2015). These meetings usually take place with all team members or representatives, so that they can discuss issues about the project, reduce the sociocultural distance, improve team collaboration and communication. These visits help to build relationships and trust (RAJPAL, 2018; Hossain; Babar; Paik, 2009; Nuevo; Piattini; Pino, 2011), and, according to Szabó and Steghofer, (Szabó; Steghöfer, 2019) they make it possible to identify and solve problems collaboratively (LOUS; KUHRMANN; TELL, 2017; Paasivaara, 2017; HOSSAIN, 2019; VALLON et al., 2017; MOE et al., 2015; RIZVI; BAGHERI; GASEVIC, 2015; VALLON et al., 2013; RALPH; SHPORTUN, 2013; DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; LASSENIUS, 2010; AVRITZER; BRONSARD; MATOS, 2010; HOSSAIN; BABAR; VERNER, 2009b; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; HOLE; MOE, 2008; LEE; YONG, 2009).

4.2.6 Retrospective meeting (19)

It is an ”improvement“ meeting held to find ways and means of identifying possible pitfalls, past mistakes, and looking for new ways to avoid those mistakes (Sundararajan; Bhasi; Vijayaraghavan, 2014). *Goal*: a retrospective meeting is held to check and improve the project's execution processes. *Who*: dev teams and scrum master. *How*: in GSD, retrospective meetings are held through telephones, videoconferences, webcam, and other communication tools (Robinson, 2019; HOSSAIN; BANNERMAN; JEFFERY, 2011b; MOE et al., 2015). Some of the retrospectives were carried out using synchronous communication tools (Szabó; Steghöfer, 2019), while others, as mentioned in the studies (KAUSAR; AL-YASIRI, 2017; Hossain; Babar; Paik, 2009), were carried out asynchronously, through the publication of comments and Wiki results and information shared by email (LOUS; KUHRMANN; TELL, 2017; S.; KUMAR; MANI, 2018; VALLON et al., 2017; TRIPATHI et al., 2015; RIZVI; BAGHERI; GASEVIC, 2015; Sundararajan; Bhasi; Vijayaraghavan, 2014; DORAIRAJ; NOBLE; MALIK, 2012; HOSSAIN; BANNERMAN; JEFFERY, 2011a; PAASIVAARA; LASSENIUS, 2010; HOSSAIN; BABAR; VERNER, 2009a; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; DINGSøYR et al., 2017; Jalali; Wohlin, 2010).

4.2.7 Sprint (19)

Many projects from reviewed studies related the use of sprints (RIZVI; BAGHERI; GASEVIC, 2015; RALPH; SHPORTUN, 2013; AVRITZER; BRONSARD; MATOS, 2010), most of them respecting a duration from one to four weeks, but one study reported the conduction of a sprint with six weeks for a period (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). *Goal*: develop pre-defined requirements along a time-boxed period, not accepting changes during the period and producing a software increment by the end. *Who*: teams. *How*: the sprint is implemented through the application of more agile practices, such as sprint planning, sprint review, sprint retrospective (HOSSAIN; BANNERMAN; JEFFERY, 2011a). Also, some papers suggest to keep sized short sprints cycles to constantly delivery value to the customer (Szabó; Steghöfer, 2019; LOUS; KUHRMANN; TELL, 2017; HOSSAIN; BANNERMAN; JEFFERY, 2011b; LAL; CLEAR, 2018; ALSAQAF; DANEVA; WIERINGA, 2017; VALLON et al., 2017; Gupta; Manikreddy, 2015; RIZVI; BAGHERI; GASEVIC, 2015; ESTáCIO; PRIKLADNICKI, 2014; RALPH; SHPORTUN, 2013; Sriram; Mathew, 2012; RAMESH; MOHAN; CAO, 2012; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; PAASIVAARA; LASSENIUS, 2010; AVRITZER; BRONSARD; MATOS, 2010; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; HOLE; MOE, 2008; Jalali; Wohlin, 2010).

4.2.8 Product backlog (18)

The product backlog practice is popular in distributed software development, and it was seen in many articles (VALLON et al., 2017; Gupta; Manikreddy, 2015; MOE et al., 2015). It was observed that some distributed projects used electronic tools to manage product backlog (Szabó; Steghöfer, 2019; KAUSAR; AL-YASIRI, 2017; PAASIVAARA; LASSENIUS, 2010). *Goal*: aims to concentrate all the business and architectural features that need to be implemented by all distributed teams. *Who*: product owner and team lead. *How*: according to Hossain *et al.* (HOSSAIN; BANNERMAN; JEFFERY, 2011b) study, in one of the studied projects, the product backlog was refined and incremented biweekly with the customer. Lee and Yong (LEE; YONG, 2009) showed that the studied distributed teams had their own backlogs, and there were separated according to local and regional customizations (Szabó; Steghöfer, 2019; LOUS; KUHRMANN; TELL, 2017; GUPTA; JAIN; SINGH, 2018; LAL; CLEAR, 2018; RICHTER; RAITH; WEBER, 2016; ALSAQAF; DANEVA; WIERINGA, 2017; VALLON et al., 2017; Khmelevsky; Li; Madnick, 2017; KAUSAR; AL-YASIRI, 2017; Gupta; Manikreddy, 2015; MOE et al., 2015; RALPH; SHPORTUN, 2013; LI; MäDCHE, 2013; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; PAASIVAARA; LASSENIUS, 2010; Nuevo; Piattini; Pino, 2011).

4.2.9 Backlog management (16)

Backlog management is the technique used by agile teams to record, track, and prioritize what needs to be implemented in the project so that the goals are met (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; AVRITZER; BRONSARD; MATOS, 2010). *Goal*: ensure the organization of the tasks to be performed. *Who*: product owner and teams. *How*: it is important to ensure that team members always have a clear view of the product (MOE et al., 2015; LEE; YONG, 2009). In this way, the developers, according to Paasivaara, Durasiewicz, and Lassenius (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008) used a support tool called Jira [1] to manage the backlog and make it available to anyone in the distributed teams (S.; KUMAR; MANI, 2018; LAL; CLEAR, 2018; RAMESH et al., 2006; Hamid, 2013; VALLON et al., 2013; LI; MäDCHE, 2013; HOSSAIN; BANNERMAN; JEFFERY, 2011a; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; MARUPING, 2010; AVRITZER; BRONSARD; MATOS, 2010; HOLE; MOE, 2008; DINGSøYR et al., 2017; Jalali; Wohlin, 2010).

4.2.10 User stories (15)

The user stories are used in distributed projects for the same purpose as co-located ones. It describes, in a detailed way, the needs of the project according to

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

the user's vision, thus representing a system requirement (MODI; ABBOTT; COUNSELL, 2017; TRIPATHI et al., 2015; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). *Goal*: guarantee an understanding of functional and non-functional requirements by providing the vision of the end-user to team members. *Who*: product owner and teams. *How*: according to Alsaqaf, Daneva, and Wieringa (ALSAQAF; DANEVA; WIERINGA, 2017) in the global context, user stories can be distributed according to the nature of the story and the skills available in the distributed teams. Other studies registered user stories in web platforms, such as Jira, make them available among all team members and modified as necessities arose, thus ensuring coordinated work among developers (Szabó; Steghöfer, 2019; LOUS; KUHRMANN; TELL, 2017; S.; KUMAR; MANI, 2018; LAL; CLEAR, 2018; RAMESH et al., 2006; VALLON et al., 2017; MODI; ABBOTT; COUNSELL, 2017; TRIPATHI et al., 2015; RIZVI; BAGHERI; GASEVIC, 2015; Sundararajan; Bhasi; Vijayaraghavan, 2014; TANNER; CHIGONA, 2012; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; Jalali; Wohlin, 2010; Hossain; Babar; Paik, 2009).

4.2.11 Pair programming (15)

The practice of pair programming was quite popular among the reviewed studies, most of the projects from those studies applied pair programming between team members of the same site (Szabó; Steghöfer, 2019; SCHENK; PRECHELT; SALINGER, 2014; HOSSAIN, 2019). Although, some studies applied distributed pair programming (DPP) between team members of different sites through screen sharing and video conference (LOUS et al., 2018a; ESTáCIO; PRIKLADNICKI, 2014). *Goal*: develop the source code of a project through two developers working on the same computer, aiming to improve the quality of the code and share knowledge. *Who*: dev teams. *How*: according to some studies (SCHENK; PRECHELT; SALINGER, 2014; RAZZAK et al., 2017; ESTáCIO; PRIKLADNICKI, 2014) is important to select proper communication tools which allow audio and video sharing for the application of distributed pair programming (DPP) (Szabó; Steghöfer, 2019; LOUS et al., 2018a; LAL; CLEAR, 2018; HOSSAIN, 2019; VALLON et al., 2017; KAUSAR; AL-YASIRI, 2017; Gupta; Manikreddy, 2015; TRIPATHI et al., 2015; RIZVI; BAGHERI; GASEVIC, 2015; MARUPING, 2010; AVRITZER; BRONSARD; MATOS, 2010; Jalali; Wohlin, 2010).

4.2.12 Sprint review (14)

Sprint review was one of the most popular practices found in the review (LOUS; KUHRMANN; TELL, 2017; HOSSAIN; BANNERMAN; JEFFERY, 2011b; VALLON et al., 2013). The review meeting in distributed teams is commonly conducted together with a system demo (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; VALLON et al., 2013). *Goal*: present the software increment to the product owner, and the pro-

ject stakeholders aiming to receive feedback about the features developed. *Who*: teams, product owner, scrum master, and stakeholders. *How*: the sprint review meetings are usually conducted via video or phone conferences (KAUSAR; AL-YASIRI, 2017). However, the reviewed studies differ a lot of whom should be present in the review meeting, suggesting only the presence of the PO (Szabó; Steghöfer, 2019), or the presence of scrum master, PO and project stakeholders (LOUS; KUHRMANN; TELL, 2017; HOSSAIN; BANNERMAN; JEFFERY, 2011b; VALLON et al., 2017; Khmelevsky; Li; Madnick, 2017; MOE et al., 2015; RIZVI; BAGHERI; GASEVIC, 2015; VALLON et al., 2013; TANNER; CHIGONA, 2012; HOSSAIN; BANNERMAN; JEFFERY, 2011a; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; HOSSAIN; BABAR; VERNER, 2009a; LEE; YONG, 2009; Jalali; Wohlin, 2010).

4.2.13 Self-management (13)

Agile distributed projects adopted a self-management approach for their environment. Moving the main management activities and work-related activities to the team members, effectively eliminating the traditional management structure (GUPTA; JAIN; SINGH, 2018; RAZZAK et al., 2017; MODI; ABBOTT; COUNSELL, 2017). *Goal*: align priorities, plans, keep the team more focused around work in a responsible way. *Who*: teams. *How*: several papers (RAMESH et al., 2006; RIZVI; BAGHERI; GASEVIC, 2015; Sundararajan; Bhasi; Vijayaraghavan, 2014) argued that self-management needs to apply through allowing everyone to choose their tasks, focusing on continuous learning activities, sharing knowledge and team building, being able to share and define problems and activities when possible, and assigning unique responsibilities (GUPTA; JAIN; SINGH, 2018; LAL; CLEAR, 2018; ALSAQAF; DANEVA; WIERINGA, 2017; Gervigny; Nagowah, 2017; RAZZAK et al., 2017; RAZZAK, 2016b; Gupta; Manikreddy, 2015; MOE et al., 2015; TRIPATHI et al., 2015; HUBER; DIBBERN, 2014).

4.2.14 Continuous integration (11)

Many reviewed studies presented projects that applied continuous integration (LOUS; KUHRMANN; TELL, 2017; Sundararajan; Bhasi; Vijayaraghavan, 2014; HOLE; MOE, 2008) as a practice of distributed teams. One project related having a specific distributed team to handle all activities regarding continuous integration (Szabó; Steghöfer, 2019). *Goal*: it aims to ensure that the value is being delivered to the customer constantly through an automated process. *Who*: dev teams. *How*: a study reported that solution modules were integrated overnight to identify build failures (Sundararajan; Bhasi; Vijayaraghavan, 2014). Furthermore, use optional regression tests and failure reports can be helpful to identify any issue that could come from a failure to build (Szabó; Steghöfer, 2019; LOUS; KUHRMANN; TELL, 2017; VALLON et al., 2017; RIZVI; BAGHERI; GA-

SEVIC, 2015; Sriram; Mathew, 2012; AVRITZER; BRONSARD; MATOS, 2010; HOLE; MOE, 2008; LEE; YONG, 2009; Jalali; Wohlin, 2010; Nuevo; Piattini; Pino, 2011).

4.2.15 Burndown charts (11)

In a distributed environment the use of electronic tools to support the burndown charts seems to be the best option for teams (HOSSAIN; BANNERMAN; JEFFERY, 2011b; RICHTER; RAITH; WEBER, 2016; KAUSAR; AL-YASIRI, 2017; LEE; YONG, 2009). For instance, burndown is the most common and shows the estimated amount of remaining sprint backlog across the time of a sprint (Nuevo; Piattini; Pino, 2011). *Goal*: discover the capacity of the team, showing the members their progress, and their estimation accuracy. *Who*: scrum master. *How*: according to the studies (HOSSAIN; BANNERMAN; JEFFERY, 2011b; RICHTER; RAITH; WEBER, 2016; KAUSAR; AL-YASIRI, 2017; LEE; YONG, 2009), the best way to implement burndown charts in distributed teams is by using electronic tools. It makes the charts available to any member across the globe and helps synchronize the teams (VALLON et al., 2017; VALLON et al., 2013; RALPH; SHPORTUN, 2013; TANNER; CHIGONA, 2012; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; Jalali; Wohlin, 2010; Nuevo; Piattini; Pino, 2011).

4.2.16 Synchronize work hours (10)

The synchronization of working hours is a practice used in distributed agile development in which teams that have different time zones seek to have, at least, some intersection in working time(Hossain; Babar; Paik, 2009). *Goal*: increase communication and collaboration among team members, and reduce misunderstandings with synchronized working hours. *Who*: teams. *How*: The synchronization of working hours can vary, as the variations in time zones can be very different. Thus, it is necessary to find common times, so that teams can have synchronous communication (Sundararajan; Bhasi; Vijayaraghavan, 2014; HOSSAIN; BANNERMAN; JEFFERY, 2011a; BASS, 2015). Besides it, according to Nuevo *et al.*, (Nuevo; Piattini; Pino, 2011) the meetings hours can be alternated, like taking meetings during a team's normal working hours and at another time conduct the meetings during the other team's hours (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; HOSSAIN; BABAR; VERNER, 2009b; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; KUSSMAUL; JACK; SPONSLER, 2004; LEE; YONG, 2009; Hossain; Babar; Paik, 2009).

4.2.17 Coaching (9)

The studies related to coaching focused on teaching the agile methods to the client (RIZVI; BAGHERI; GASEVIC, 2015), training the globally distributed teams for the agile process (LEE; YONG, 2009), having meetings to reinforce the value of scrum

(Hossain; Babar; Paik, 2009), and training the team members for the application of adapted practices such distributed programming(ESTáCIO; PRIKLADNICKI, 2014) and on coding skills (MOE et al., 2015). *Goal*: it aims to coach the distributed teams, and the customer in agile methods to align them with the agile process. Also, it provides training for the team members in technical skills. *Who*: scrum master and team lead. *How*: according to Rizvi *et al.* (RIZVI; BAGHERI; GASEVIC, 2015), the scrum master was responsible to coach the organization into the agile way of working. Furthermore, some companies provide training in agile methods for the globally distributed teams (LEE; YONG, 2009) and others send their more experienced people to another site for training the junior members (S.; KUMAR; MANI, 2018; MOE et al., 2015; ESTáCIO; PRIKLADNICKI, 2014; Sundararajan; Bhasi; Vijayaraghavan, 2014; DUMITRIU; OPREA; MESNITA, 2011; HOSSAIN; BABAR; VERNER, 2009b; Hossain; Babar; Paik, 2009).

4.2.18 Task management (9)

Task management is a practice that manages a task throughout its process. Its focus on assigning tasks to the team members according to their roles and skills, and to make them visible to anyone in the distributed sites (KAUSAR; AL-YASIRI, 2017; BANIJAMALI et al., 2017). *Goal*: help in team collaboration and coordination throughout the execution of project tasks. *Who*: teams, product owner, and scrum master. *How*: Kausar and Yasiri found in the study (KAUSAR; AL-YASIRI, 2017) that the practice is used through online sharing tools, in which the entire team can have access to the product backlog, storyboard, taskboard, burndown graphics, and other agile artifacts. In addition, a central code repository was used so that teams can see the progress of the tasks (HOSSAIN; BANNERMAN; JEFFERY, 2011b; LOUS et al., 2018a; BANIJAMALI et al., 2017; MODI; ABBOTT; COUNSELL, 2017; CHO, 2007; KUSSMAUL; JACK; SPONSLER, 2004; Hossain; Babar; Paik, 2009).

4.2.19 Necessary documentation (9)

The necessary documentation is the practice of producing the minimum amount of document in the agile software development that generates value (RIZVI; BAGHERI; GASEVIC, 2015; Hossain; Babar; Paik, 2009). *Goal*: improve communication and make information regarding the project available to everybody. *Who*: teams. *How*: according to Hossain *et al.* study, (Hossain; Babar; Paik, 2009), the use of web platforms, such as Jira, makes it easier to share the information across distributed teams online and to track the problems documented. Also, it was observed in the study (RIZVI; BAGHERI; GASEVIC, 2015) that, by creating documentation, the team members reduced the constant need for communication, which worked as a benefit once communication among teams from different cultures can cause problems (RICHTER; RAITH; WEBER, 2016;

Sundararajan; Bhasi; Vijayaraghavan, 2014; Hamid, 2013; AVRITZER; BRONSARD; MATOS, 2010; HOSSAIN; BABAR; VERNER, 2009b; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; Nuevo; Piattini; Pino, 2011).

4.2.20 Kanban board (8)

Some studies showed that each distributed team had its physical kanban board on their sites (RIZVI; BAGHERI; GASEVIC, 2015; VALLON et al., 2013; DINGSøYR et al., 2017). However, other studies presented distributed teams that used electronic boards (HUBER; DIBBERN, 2014; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011), and teams that used to have both electronic and physical Kanban boards (TRIPATHI et al., 2015). *Goal*: promote the visualization and the limits of the work in process in distributed teams across different sites. *Who*: teams, product owner, scrum master. *How*: according to the studies reviewed, each distributed team can have their physical kanban board in their sites (TRIPATHI et al., 2015; VALLON et al., 2013; DINGSøYR et al., 2017), but it can make it hard to replicate the status among the different sites. Also, it is possible to use an electronic board for all distributed teams which makes it easy for any team to visualize the status updates of tasks (HUBER; DIBBERN, 2014; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). Furthermore, scaling Kanban was a practice used by organizations throughout its structure through training and coaching, to face challenges during large-scale distributed projects (LAL; CLEAR, 2018; TRIPATHI et al., 2015; RIZVI; BAGHERI; GASEVIC, 2015; Jalali; Wohlin, 2010).

4.2.21 Design the team (8)

Some articles suggest to disperse the most technical members across different distributed teams (HOSSAIN; BABAR; VERNER, 2009b; Hossain; Babar; Paik, 2009; Nuevo; Piattini; Pino, 2011) and to form teams with members located across different time zones to enhance the development across 24/7 routine (MARUPING, 2010; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). *Goal*: form cross-functional agile distributed teams capable of achieving the project's success with their members. *Who*: product owner and scrum master. *How*: one study showed a distributed project that had a restriction for team distribution, in the project there was multiple distributed sites, but an agile distributed team could only have members between two different sites (Hossain; Babar; Paik, 2009). Furthermore, the study of Vallon *et al.* presented a distributed project with three scrum teams formed across all products and based on logical requirements areas (GUPTA; JAIN; SINGH, 2018; VALLON et al., 2013; LI; MäDCHE, 2013; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; MARUPING, 2010; HOSSAIN; BABAR; VERNER, 2009b; Nuevo; Piattini; Pino, 2011).

4.2.22 Co-locate all team members at the beginning (8)

Get the whole teams together at the beginning of the project can be helpful to exchange experiences, project goals with the team members (RALPH; SHPORTUN, 2013; DORAIRAJ; NOBLE; MALIK, 2012; PAASIVAARA; LASSENIUS, 2010). *Goal*: facilitate team interaction even for a short time. It aims to establish trust in the entire team. *Who*: teams. *How*: according to the reviewed studies (AVRITZER; BRONSARD; MATOS, 2010; Hossain; Babar; Paik, 2009; Nuevo; Piattini; Pino, 2011), the agile distributed projects choose to gather all the team members, when possible, at the beginning of the project. It aims to execute the first sprints with the team members co-located, improving their relationship, and sharing the project vision among them (RIZVI; BAGHERI; GASEVIC, 2015; RALPH; SHPORTUN, 2013; DORAIRAJ; NOBLE; MALIK, 2012; HOSSAIN; BANNERMAN; JEFFERY, 2011a; PAASIVAARA; LASSENIUS, 2010).

4.2.23 Test driven development - TDD (7)

TDD is a software development practice in which tests are written before the features are ready so that the code produced is made to pass these tests and to avoid waste in coding (Nuevo; Piattini; Pino, 2011). *Goal*: guarantee quality, avoid code waste, and improve the development process. *Who*: dev teams. *How*: according to Hossain, Babar, and Verner (HOSSAIN; BABAR; VERNER, 2009a), TDD allowed for a standardized view of development, which facilitated a better understanding of what functionality was required under the customer's view (HOSSAIN; BABAR; VERNER, 2009a). Furthermore, one study suggested developing only the code needed to run all the tests, and whether the tests are successful, the next step would be refactoring the code (LAL; CLEAR, 2018; VALLON et al., 2017; TRIPATHI et al., 2015, 2015; Jalali; Wohlin, 2010; Nuevo; Piattini; Pino, 2011).

4.2.24 Project wiki (7)

Information regarding distributed projects needs to be available to anyone in different sites, because of it, an important practice followed by agile distributed teams is the building of a project wiki (LEE; YONG, 2009; Hossain; Babar; Paik, 2009; Nuevo; Piattini; Pino, 2011). *Goal*: concentrate all the information regarding the distributed project in one place that can be accessible for any team member. *Who*: teams. *How*: the project wiki is usually used to store information regarding system documentation, functional tests, architectural guidelines, team routines, and sprint retrospectives (DINGSøYR et al., 2017). Furthermore, studies showed that agile distributed teams usually have a collaborative online wiki to store the project documentation (AVRITZER; BRONSARD;

MATOS, 2010; HOSSAIN; BABAR; VERNER, 2009a; HOSSAIN; BABAR; VERNER, 2009b; LEE; YONG, 2009; Hossain; Babar; Paik, 2009; Nuevo; Piattini; Pino, 2011).

4.2.25 Estimation meeting (6)

Estimation meetings are events that take place among distributed teams during a project to evaluate the effort needed to develop the requirements for that sprint (VALLON et al., 2017). *Goal*: measure the necessary effort to develop the project tasks and meet the delivery deadlines. *Who*: dev teams. *How*: the reviewed studies do not describe properly how to conduct estimation meetings. However, as many of the distributed meetings, it needs to be through web-based platforms, such as videoconference, audioconferences. However, the studies described the use of many estimation techniques, like task size, story points, use case points, etc (VALLON et al., 2017; GONÇALVES et al., 2017; Britto; Mendes; Börstler, 2015; TRIPATHI et al., 2015; BRITTO; USMAN; MENDES, 2014; Sundararajan; Bhasi; Vijayaraghavan, 2014).

4.2.26 Continuous deployment (6)

In the reviewed studies, continuous deployment was cited as a practice of some agile distributed teams (RIZVI; BAGHERI; GASEVIC, 2015; HOSSAIN; BABAR; VERNER, 2009b; HOLE; MOE, 2008). However, as discussed in one of the studies (HOSSAIN; BABAR; VERNER, 2009b) the everyday deployment of code in the production environment by a non-experienced team was bad, because it came with a large number of bugs. *Goal*: ensure that staged features are constantly deployed in the production server through an automated process of integration tests and deployment. *Who*: dev teams. *How*: the studies reviewed do not present a definitive way of applying continuous deployment. One company enhanced its continuous deployment process by using CruiseControl[2] to rebuild the project when an update was made though (LAL; CLEAR, 2018; RIZVI; BAGHERI; GASEVIC, 2015; AVRITZER; BRONSARD; MATOS, 2010; HOSSAIN; BABAR; VERNER, 2009b; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; HOLE; MOE, 2008) .

4.2.27 System demo (6)

It is the method for assessing the solution's current state and gathering immediate feedback from the product owner, stakeholders, and customers (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; DINGSøYR et al., 2017). *Goal*: demonstrate the developed requirements of the last iteration. It aims to receive feedback about the work done. *Who*: product owner, dev teams, scrum master, and stakeholders. *How*: accor-

[2] cruisecontrol.sourceforge.net

ding to Passivara *et al.* (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008), systems demos were organized through teleconferencing and application sharing, with the participation of the team members, the product owner, and the scrum master (LAL; CLEAR, 2018; Sundararajan; Bhasi; Vijayaraghavan, 2014; DINGSøYR et al., 2017; Jalali; Wohlin, 2010; Hossain; Babar; Paik, 2009).

4.2.28 Test automation (6)

It is used to automate repetitive tasks and testing tasks which are difficult to perform manually. It aims to ensure that built versions of the solution are working ok (HOSSAIN; BABAR; VERNER, 2009b; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008). *Goal*: test automation aims to execute many test cases automatically and repeatedly when a new version of the project is deployed. *Who*: dev teams. *How*: according to Passivara *et al.*, (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008) case study, the developed product used to be built every night with a set of automated tests, and whether a built was unsuccessful the distributed team responsible for this building would be responsible for the fix (LOUS; KUHRMANN; TELL, 2017; VALLON et al., 2017; RAZZAK et al., 2017; HOSSAIN; BABAR; VERNER, 2009b; Jalali; Wohlin, 2010).

4.2.29 Code review (6)

It is a software development practice in which one or more developers check parts of the developed code and correct defects found. It aims to guarantee product quality and share knowledge among team members (MOE et al., 2015; Hamid, 2013; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). *Goal*: improve code quality, detect bugs early, share knowledge, reduce tests, and assure the solution quality. *Who*: dev teams. *How*: Rizvi *et al.* study (RIZVI; BAGHERI; GASEVIC, 2015) presents that senior resources can assist the code review due to their skills. Furthermore, one study suggests the senior developers perform the code review together with junior developers (MOE et al., 2015). It aims to coach the junior members to write better code in the desired standards (Hamid, 2013; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; HOLE; MOE, 2008; Jalali; Wohlin, 2010).

4.2.30 Collaboration among teams (6)

The practice aims to reinforce the collaborations and relationships among the team members (DORAIRAJ; NOBLE; MALIK, 2011). Also, some unofficial meetings were conducted for distributed teams just have discussed personal matters and have some kind of fun conversation (DORAIRAJ; NOBLE; MALIK, 2012). *Goal*: it aims to increase team collaboration, generating trust, and creating a pleasant atmosphere. *Who*: teams. *How*: according to some studies, the collaboration sessions can be made before

an official meeting, to clarify issues, share problems, or just socialize (Paasivaara, 2017; DORAIRAJ; NOBLE; MALIK, 2012; TANNER; CHIGONA, 2012; DORAIRAJ; NOBLE; MALIK, 2011; DINGSøYR et al., 2017; Hossain; Babar; Paik, 2009).

4.2.31 Manage customer expectations (6)

As the name says it is a way to manage customer expectations about the work in progress. The use of short iterations, frequent feedbacks, and well-managed backlog are powerful techniques for handling customer expectations (AVRITZER; BRONSARD; MATOS, 2010). *Goal*: keep the client aware of what is happening in the project, maintaining communication and visibility of the project progress. *Who*: product owner. *How*: it is necessary to keep frequent contact with the customer to manage their expectations (MOE et al., 2015; RAMESH; MOHAN; CAO, 2012), and make them part of the team (DORAIRAJ; NOBLE; MALIK, 2012). Furthermore, share the backlog content and progress of the solution to the client can help in the expectation management (VALLON et al., 2017; AVRITZER; BRONSARD; MATOS, 2010; Jalali; Wohlin, 2010).

4.2.32 Planning game (6)

According to Rizvi study *et al.* (RIZVI; BAGHERI; GASEVIC, 2015) the planning game aims to get prioritized requirements from the customer, and then, estimate it. *Goal*: plan the prioritized tasks by estimating them with game techniques. *Who*: dev teams, product owner, and scrum master. *How*: according to Maruping study (MARUPING, 2010), it is more effective to hold the plan game through synchronous communication, such as videoconference or telephone. Furthermore, the planning poker technique appeared as a common technique for planning games in distributed teams (VALLON et al., 2017; KAUSAR; AL-YASIRI, 2017; RIZVI; BAGHERI; GASEVIC, 2015; BRITTO; USMAN; MENDES, 2014; Jalali; Wohlin, 2010).

4.2.33 Continuous delivery (6)

The continuous delivery practice consists of delivering code frequently, like daily, weekly, or every sprint interaction (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; KUSSMAUL; JACK; SPONSLER, 2004). Also, It aims to constantly deliver tangible value to the customer (RAMESH et al., 2006). *Goal*: deliver software increments to the customer frequently. *Who*: dev teams. *How*: the studies reviewed mentioned the code deliveries as a frequent activity whose schedule can be defined based on the project needs (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; KUSSMAUL; JACK; SPONSLER, 2004). However, one study suggests the delivery of values since the early phases of the project (RAJPAL, 2018; LAL; CLEAR, 2018; RAMESH et al., 2006; HOLE; MOE, 2008).

4.2.34 Assign a role to each project member (5)

During the review, we found some articles that described with full details their agile team roles (Sundararajan; Bhasi; Vijayaraghavan, 2014; VALLON et al., 2013; RALPH; SHPORTUN, 2013). Most of them used a hybrid team roles approach due to the distributed environment they were dealing, and the size of the teams. *Goal*: assign a role to each project member from the distributed teams based on their existing skills. *Who*: team lead. *How*: the team roles of a distributed agile development team should be assigned to the project members based on their existing skills and knowledge. As saw in (Gupta; Manikreddy, 2015) the focus should be given to form a cross-functional team with all the necessary expertise to develop the solution (Sundararajan; Bhasi; Vijayaraghavan, 2014; VALLON et al., 2013; RALPH; SHPORTUN, 2013; LI; MäDCHE, 2013).

4.2.35 Agile architecture (5)

The agile architecture supports the application of agile practices by evaluating the impact of user stories in the component architecture (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). *Goal*: guarantee the architecture design simplicity of the solution while the agile process is conducted. *Who*: team lead and architects. *How*: agile architecture can be used to remove inefficiencies from the software development process (RIZVI; BAGHERI; GASEVIC, 2015). The agile architecture meetings are commonly conducted through videoconferencing meetings between architects and team leaders to discuss improvements across the architecture and its components, to add value to the product (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; Jha; Vilardell; Narayan, 2016). Besides, in another study (Hossain; Babar; Paik, 2009), authors reported that an independent architecture with well-defined interfaces enabled teams to be more autonomous (DINGSøYR et al., 2017).

4.2.36 Coding standards (5)

Coding standards are a formalized set of rules and practices that should be followed by developers from all sites (HOSSAIN; BABAR; VERNER, 2009a; Jalali; Wohlin, 2010). *Goal*: assist developers with the formalization of common standards for writing readable and sustainable code. *Who*: team lead and dev teams. *How*: it was suggested that coding standards need to be discussed and established early in the development process (MARUPING, 2010). And the senior members of the project should disseminate the standards and guarantee that all team members had understood the rules that will guide the development. Furthermore, the coding standards need to be followed by all distributed sites (VALLON et al., 2017; HOSSAIN; BABAR; VERNER, 2009a; LEE; YONG, 2009; Jalali; Wohlin, 2010).

4.2.37 Collective code ownership (5)

The practice aims to encourage the team members that they are responsible for the system design. Some articles presented the use of collective code ownership in agile distributed teams (Szabó; Steghöfer, 2019; VALLON et al., 2017; TRIPATHI et al., 2015; LEE; YONG, 2009). *Goal*: apply collective code ownership to encourage every team member to be responsible for the system's design. Also, be able to change any line of code, add features, fix bugs, and refactor. *Who*: dev teams. *How*: according to Maruping (MARUPING, 2010), the team members need to be aware of their roles and responsibilities before the implementation of collective code ownership and to achieve this a meeting involving all sites is necessary (Szabó; Steghöfer, 2019; VALLON et al., 2017; TRIPATHI et al., 2015; LEE; YONG, 2009).

4.2.38 Refactoring (5)

Refactoring focuses on restructuring the system, removing duplications and unnecessary code, simplifying the solution design, improving communication, and providing a better understanding of the code (HOSSAIN; BABAR; VERNER, 2009a). *Goal*: simplify maintenance, improve code quality, and understanding. *Who*: dev teams. *How*: according to the case study of Szabó and Steghofer (Szabó; Steghöfer, 2019), the refactoring sessions were conducted once or twice per month in their distributed project (VALLON et al., 2017; HOSSAIN; BABAR; VERNER, 2009a; LEE; YONG, 2009; Jalali; Wohlin, 2010).

4.2.39 Frequent feedbacks (4)

Frequent feedback was seen as a practice to improve communication among distributed teams (MOE et al., 2015). Also, to avoid miscommunications or misunderstandings of requirements among the teams and the customer(RIZVI; BAGHERI; GASEVIC, 2015; AVRITZER; BRONSARD; MATOS, 2010). *Goal*: improve communication among team members and the customer, motivate the team members, and reduce miscommunications. *Who*: team lead and stakeholders. *How*: the feedback sessions for the team can be made monthly to understand how each member can improve to achieve the goals (Gupta; Manikreddy, 2015). Besides it, feedback loops can be made with the customer and the team to prevent issues in developed requirements (RIZVI; BAGHERI; GASEVIC, 2015; MOE et al., 2015; AVRITZER; BRONSARD; MATOS, 2010).

4.2.40 Bug tracking (4)

A bug tracking system is a software application that keeps track of reported software bugs in software development projects. In GSD some articles reported the

use of web applications for issue tracking (HOSSAIN; BABAR; VERNER, 2009b; CHO, 2007). *Goal*: keep track of reported bugs and information regarding them. *Who*: dev teams. *How*: several papers mentioned that they used web tools to track problems and errors, such as Jira (HOSSAIN; BABAR; VERNER, 2009b; CHO, 2007; Hossain; Babar; Paik, 2009) and Bugzilla (LEE; YONG, 2009). All of them, with the same goal of tracking the issues that emerge from the solution.

4.2.41 Documentation of lessons learned (4)

Documenting lessons learned is a practice in agile development in which developers come together to score successes and mistakes to improve teamwork in future iterations(RIZVI; BAGHERI; GASEVIC, 2015). *Goal*: improve teamwork in future iterations. *Who*: teams. *How*: according to Rizvi, Bagheri, and Gasevic (RIZVI; BAGHERI; GASEVIC, 2015), it is recommended to document the lessons learned at the end of each sprint. Furthermore, the use of lessons learned repository was reported (Sundararajan; Bhasi; Vijayaraghavan, 2014; PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; Hossain; Babar; Paik, 2009).

4.2.42 Share mission and vision (3)

The goal of the practice is to align all global team members with the goal of the solution by sharing the mission and vision of the project. In (LAL; CLEAR, 2018) the authors suggest the practice, to lead and align the developers to implement the solution considering the end-user experience instead of functionalities driven development. *Goal*: align all distributed teams to a common goal, vision, and mission for developing the solution based on feasible and valuable requirements to the users. *Who*: product owner and scrum master. *How*: the study of Gupta and Manikreddy (Gupta; Manikreddy, 2015) showed that one team member can be responsible for ensuring that all distributed teams were understanding the project vision and the “Big picture” (LAL; CLEAR, 2018; LEE; YONG, 2009).

4.2.43 Rotate team members among sites (3)

Rotate team members among sites can be helpful to improve the relationships of the team members, build trust among them, and promote the training of senior to juniors members (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008). *Goal*: rotate team members among different sites to improve peer-to-peer relationships and build trust. *Who*: team lead and scrum master. *How*: it was reported in one study that some team members rotate their location near to the customer location (DORAIRAJ; NOBLE; MALIK, 2012). Furthermore, a case study suggested the rotation of members among

teams, specifically when new members arrived and existing teams needed to be split (PAASIVAARA; DURASIEWICZ; LASSENIUS, 2008; DINGSøYR et al., 2017).

4.2.44 Simple design (3)

The simple design practice aims to keep the simplicity of the solution, especially the code and architecture simplicity. *Goal*: keep the solution design simple along the development process. *Who*: dev teams. *How*: one reviewed study suggested that the simple design is a continuous activity that aims to simplify existing work (Szabó; Steghöfer, 2019). Besides it, the same study reported simple design can be affected by sociocultural distance, leading to decision making without proper technical information (VALLON et al., 2017; LEE; YONG, 2009).

4.2.45 Roadmap Planning (3)

The roadmap planning is an agile development practice that seeks to plan in a flexible way the main stages of the project to achieve business objectives and market needs (LAL; CLEAR, 2018). *Goal*: it helps the team understand the project vision. *Who*: product owner and team lead. *How*: according to Hossain *et al.* (Hossain; Babar; Paik, 2009), it was found that the roadmap planning took place in quarterly meetings, contributing to a better understanding of the project's vision. Also, According to Lal and Tony (LAL; CLEAR, 2018), the use of roadmap planning enabled the team to make more collective decision-making, expanded the concern of time-to-market, and improved the identification of high-level requirements for short-term projects (LAL; CLEAR, 2018; TANNER; CHIGONA, 2012; Nuevo; Piattini; Pino, 2011).

4.2.46 Acceptance tests (3)

Acceptance tests are used to ensure that the developed stories are passing the acceptance criteria required by the user. *Goal*: evaluate the system's compliance according to the business requirements and customer needs. *Who*: product owner, stakeholders, and dev teams. *How*: the reviewed studies do not specify how to conduct acceptance tests, only one study reported that the acceptance criteria for delivery were defined at project startup (LAL; CLEAR, 2018; Sundararajan; Bhasi; Vijayaraghavan, 2014; Jalali; Wohlin, 2010).

4.2.47 Tests management (2)

The test management practice is related to a test plan and test report strategy (Nuevo; Piattini; Pino, 2011). The test plan is organized containing all the tests that need to be performed, and the required tools and resources for it (Nuevo; Piattini; Pino,

2011). Furthermore, the test report contains the scripts used on the test, results of the performed tests, and the resolutions of the issues (Jha; Vilardell; Narayan, 2016). *Goal*: manage the tests of the project through test plans and test reports. *Who*: testers and dev teams. *How*: according to Madan and Rosa (Jha; Vilardell; Narayan, 2016), test management can be conducted through virtual weekly meetings among distributed teams, aiming to discuss testing related progress, dependencies, and issue resolution (Nuevo; Piattini; Pino, 2011).

4.2.48 Expand teams responsibility gradually (2)

The expansion of responsibilities focuses on making the team members independent to solve complex problems (MOE et al., 2015). *Goal*: make the team members more responsible and capable of doing the project tasks independently. *Who*: team lead. *How*: in the context of GSD, the expansion of responsibilities occurs through small steps, gradually introducing the most important backlog resources for the teams (MOE et al., 2015; Nuevo; Piattini; Pino, 2011).

4.3 Discussion and Conclusion of agile practices in AGSD

In the reviewed literature, we find 48 agile practices being applied in GSD. Daily meetings (see Section 4.2.1), communication practices (see Section 4.2.2), and planning (see Section 4.2.3) were the most cited practices, respectively with 36, 34, and 24 citations. We realized that both practices have a common point, communication, which emphasizes an important value of the agile manifesto: individuals and interactions over processes and tools (BECK et al., 2001). Therefore, we can reaffirm that this value is also followed in AGSD, as daily meetings, communication practices, and planning promote the control and coordination of global teams, adding value and sharing the needed information across everyone in the project.

Despite the benefits of using agile practices in GSD, there are some challenges regarding the adaption, control, and coordination of these practices in a global environment. Synchronized work hours (see Section 4.2.16) is one of the hardest practices to be adopted since it is almost impossible to synchronize the work hours for distributed teams with big-time zone differences. Also, co-locate all team members at the beginning (see Section 4.2.22), organize visits among sites (see Section 4.2.5), and rotate team members among sites (see Section 4.2.43) are not cheap or affordable practices for all distributed projects which make these practices difficult too.

Beyond the challenges faced, it is important to point to the popularity of Scrum practices in AGSD since eight of the ten most cited practices from the reviewed studies are from Scrum, being daily meetings (see Section 4.2.1), planning (see Section 4.2.3),

scrum of scrums (see Section 4.2.4), retrospective meeting (see Section 4.2.6), sprint (see Section 4.2.7), product backlog (see Section 4.2.8), backlog management (see Section 4.2.9), user stories (see Section 4.2.10. It is possible to say that Scrum is the most successful agile method applied from 2004 to 2019 in GSD. Furthermore, the studies that reported the use of such agile practices show how adaptable they are in a global environment. Pointing that most of the meetings and events presented in agile methods can be handled with the use of online tools, web-based applications, etc (Szabó; Steghöfer, 2019; RIZVI; BAGHERI; GASEVIC, 2015; Hossain; Babar; Paik, 2009).

We observed that the number of publications became more constant since 2011 (see Figure 5). Besides it, we observed in the same graph the growing number of evaluation studies from 2014 to now, appearing more observations, surveys, and interview studies combined with case studies that can provide more substantial results. Also, it is possible to see the growth in SLR (see Figure 3) which indicates the maturity that the research area is gaining. However, it is important to say that the academy and the industry continue to work together reporting their experiences through case studies. Figure 5 confirms that in the last decade the number of case study papers continued to grow.

4.4 Which practices reported in AGSD literature embrace practices from SAFe when adopting scale agile development?

From the 19 reviewed studies related to scaling agile practices, 15 of them are case studies in real industrial scenarios as presented in table 6. We classified those 15 case studies into the taxonomy of Dingsøyr *et al.* (DINGSØYR; FÆGRI; ITKONEN, 2014) that points out the scale of agile in software development projects based on the number of teams. Based on this taxonomy we categorized four studies as large-scale projects (RAZZAK et al., 2018; S.; KUMAR; MANI, 2018; RAZZAK et al., 2017; Gupta; Manikreddy, 2015), 10 studies as very large-scale projects (Paasivaara, 2017; TRIPATHI et al., 2015; LI; MäDCHE, 2013; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; MARUPING, 2010; AVRITZER; BRONSARD; MATOS, 2010; LEE; YONG, 2009; DINGSøYR et al., 2017; Jha; Vilardell; Narayan, 2016; Nuevo; Piattini; Pino, 2011), and only one could not be classified (GUPTA; JAIN; SINGH, 2018). From those case studies three of them stated the use of SAFe, the majority reported the use of scaled agile methods, such as Scrum of scrums, Scaled Kanban, and XP.

In those studies, we could see different companies go through the process of adapting their development process to an agile large-scale level, since the adoption phase, the implementation, and maintenance of the agile process. Interestingly, the

studies present AGSD projects executed most of the time in Europe, North America, and India. Besides it, we can classify the studies based on the number of team members involved in the development, just one study had less than 100 contributors (RAZZAK et al., 2017), most of them had from more 100 to less than 1000 members (RAZZAK et al., 2018; Paasivaara, 2017; S.; KUMAR; MANI, 2018; Gupta; Manikreddy, 2015; TRIPATHI et al., 2015; LI; MäDCHE, 2013; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; MARUPING, 2010; AVRITZER; BRONSARD; MATOS, 2010; LEE; YONG, 2009; Jha; Vilardell; Narayan, 2016), and finally, one article reported an extreme large-scale case with 30 thousand members across 100 countries (Nuevo; Piattini; Pino, 2011).

Also, an overview of the scaling practices, SLR studies, and equivalent practices from SAFe are present in Table 7.

ID	Organization	Product/service	Scale	Framework(s) used	Team locations
(RAZZAK et al., 2018)	Ocuco Ltd	Optical industry	9 teams >300	SAFe	Canada, France, USA, Spain, Poland, UK, Norway, Italy, Ireland
(Paasivaara, 2017)	Comptel	Mobile usage data processing	14 teams >100	SAFe	Finland, Malaysia, Norway, UK, Bulgaria
(S.; KUMAR; MANI, 2018)	NA	Medical services	4 teams 200	Scrum of scrums and Lean	Europe, USA, India
(GUPTA; JAIN; SINGH, 2018)	NA	Mission- critical software	120	Scrum of scrums	India, Germany
(RAZZAK et al., 2017)	Ocuco Ltd	Optical industry	3 teams >70	SAFe	Europe, North America
(Gupta; Manikreddy, 2015)	NA	NA	3 teams NA	Scrum of scrums	India, Germany, Usa
(TRIPATHI et al., 2015)	NA	Telecommunications	11 teams 110	Scaled Kanban	Northern Europe, Western Europe, India
(LI; MäDCHE, 2013)	GlobCo*	Software factory	44 teams 421	Scrum of scrums	Europe, Asia
(HILLEGERSBERG; LIGTENBERG; AYDIN, 2011)	Cordys	Business Process Management (BPM)	15 teams 560	Scrum of scrums	India, Netherlands, Germany, UK, China, Americas
(MARUPING, 2010)	NA	Business Process Management (BPM)	73 teams 689	Scaled XP	India, USA
(AVRITZER; BRONSARD; MATOS, 2010)	NA	Control and monitoring of systems	25 teams 200	Scrum of scrums	USA, India, Germany, USA, Greece
(LEE; YONG, 2009)	Yahoo!	My Yahoo!	21 teams	Scrum of scrums	17 countries: Europe, North America, South America, Asia
(DINGSøYR et al., 2017)	Pension Fund*, Accenture, Steria	Office automation system	12 teams 175	Scrum of scrums	Norway
(Jha; Vilardell; Narayan, 2016)	Siemens	Platform for future building technology applications	16 teams >100	Scrum of scrums	Europe, Asia, North America
(Nuevo; Piattini; Pino, 2011)	NA	Software factory	>30.000	Mixed/custom frameworks	100 countries worldwide

Tabela 6 – Case studies

4.4.1 Communication practices

Communication practices were seen in most of the studies. The “multiple communication modes” (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; BASS, 2015) combine a variety of synchronous and asynchronous approaches. Synchronous meetings simulate face-to-face communication (RAZZAK et al., 2018; KAUSAR; AL-YASIRI, 2017; MARUPING, 2010; AVRITZER; BRONSARD; MATOS, 2010; Nuevo; Piattini; Pino, 2011). For reviewing and planning meetings, the use of screen share (KAUSAR; AL-YASIRI, 2017; BASS, 2015), videoconference, chat, and teleconference tools (BASS, 2015; Nuevo; Piattini; Pino, 2011) were also reported. However, asynchronous information transfer was also present and conducted in AGSD projects due to the time diffe-

References	AGSD Practice	SAFe Practice
(RAZZAK et al., 2018; KAUSAR; AL-YASIRI, 2017; Gupta; Manikreddy, 2015; Razavi; Ahmad, 2014; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; MARUPING, 2010; AVRITZER; BRONSARD; MATOS, 2010; BASS, 2015; Nuevo; Piattini; Pino, 2011)	Communication practices	• Constant communication • Inter-team communication
(S.; KUMAR; MANI, 2018; GUPTA; JAIN; SINGH, 2018; KAUSAR; AL-YASIRI, 2017; Gupta; Manikreddy, 2015; LI; MäDCHE, 2013; AVRITZER; BRONSARD; MATOS, 2010; LEE; YONG, 2009; DINGSøYR et al., 2017; Nuevo; Piattini; Pino, 2011)	Scrum of scrums	• Scrum of scrums
(Paasivaara, 2017; Gupta; Manikreddy, 2015; AVRITZER; BRONSARD; MATOS, 2010; LEE; YONG, 2009; Nuevo; Piattini; Pino, 2011)	Visits among sites	• Gemba walks
(TRIPATHI et al., 2015; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; DINGSøYR et al., 2017)	Kanban board	• Program and Solution Kanban • Limit work in process • WIP limits by the team • Progress of items • Buffers in WIP states • Items expedition • Program Epic Kanban if necessary
(KAUSAR; AL-YASIRI, 2017)	Task management	• Development process visibility
(S.; KUMAR; MANI, 2018; LEE; YONG, 2009)	Agile Coaching	• Cross training
(AVRITZER; BRONSARD; MATOS, 2010; Nuevo; Piattini; Pino, 2011)	Document necessary information	• Readiness activities • Vision presentation continuously
(HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; BASS, 2015; LEE; YONG, 2009; Nuevo; Piattini; Pino, 2011)	Synchronize work hours	• Common cadence and synchronization
(AVRITZER; BRONSARD; MATOS, 2010; LEE; YONG, 2009; DINGSøYR et al., 2017; Nuevo; Piattini; Pino, 2011)	Project wiki	• A solid log infrastructure • PI objectives definition • Wiki for remote teams
(DINGSøYR et al., 2017)	System demo	• System demo
(Gupta; Manikreddy, 2015; LI; MäDCHE, 2013)	Assign a role to each project member	• Agile teams • Leadership roles
(Paasivaara, 2017; DINGSøYR et al., 2017)	Collaboration among teams	• Collaboration among teams • Collaboration and organization • Collaboration between team and PO • Features definition • Collaboration between Solution and System Architects • Agile teams and business owners proximity
(HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; DINGSøYR et al., 2017; Jha; Vilardell; Narayan, 2016)	Agile architecture	• Agile architecture • Modular architecture • Intentional architecture • Architecture incrementally • Architectural Runaway
(MARUPING, 2010; LEE; YONG, 2009)	Coding standards	• Coding standards • Modular architecture • SOLID • Design patterns
(Gupta; Manikreddy, 2015; LEE; YONG, 2009)	Share mission and vision	• Top ten features for the next PI • Holistic and cohesive vision • Vision presentation continuously • Share mission and vision
(LEE; YONG, 2009)	Simple design	• Quality of architecture and design • Design patterns • Set-Based Design (SBD) • Design choices • Emergent design
(Nuevo; Piattini; Pino, 2011)	Roadmap Planning	• Roadmap • Plan as short and as flexible as possible • Quarters planning • Continuous exploration
(Nuevo; Piattini; Pino, 2011)	Expand teams responsibility gradually	• Stop-the-line-mentality

Tabela 7 – Scaling practices

rence among teams (KAUSAR; AL-YASIRI, 2017), such as use of a wiki, document repositories in AGSD articles (AVRITZER; BRONSARD; MATOS, 2010).

Similar to the practices reported in the reviewed studies, constant communication and collaboration is a SAFe practice to empower decision-making and help teams meet their responsibilities.

It is possible to link communication practices from the SLR with both practices from SAFe. Although in AGSD, due to the geographic distribution, it is more likely to happen through electronic tools, such as videoconference, chats, and emails.

4.4.2 Scrum of Scrums

It is possible to state that Scrum of Scrums (SoS) practice from AGSD studies and SAFe have almost the same goals and procedures. Both bring together the scrum master of each team to review, discuss, and coordinate the project progress, the dependencies across the teams, the impediments, and other issues (S.; KUMAR; MANI, 2018; KAUSAR; AL-YASIRI, 2017; Gupta; Manikreddy, 2015).

4.4.3 Visits among sites

Visits among sites is an AGSD practice that aims to make team members visit other team members in different sites to reduce sociocultural distances, improve communication, and achieve better collaboration among teams (Gupta; Manikreddy, 2015; Nuevo; Piattini; Pino, 2011). This AGSD practice is a bit similar to *Gemba Walks* from SAFe that suggest visits to the site where the customer work is done (Leffingwell, Dean, 2020). It aims to build empathy for the teams by providing a deeper understanding of the user's emotional and physical needs (Leffingwell, Dean, 2020). Due to the similarity of promoting visits across sites, both practices were linked.

4.4.4 Kanban board

The use of Kanban board is common in the AGSD projects and in SAFe. As we saw in the reviewed studies, some teams have their board at their sites (DINGSøYR et al., 2017), and others have presented electronic boards that were shared across different sites for all global teams (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011). Due to this, we can link the kanban board practice to the practices of team, program and solution kanban (Leffingwell, Dean, 2020) from SAFe. The team board is built by each team to visualize their current process flow. Besides, different from work in progress from agile methods, SAFe has work in process (WIP) limits that are defined by the team at the beginning (Leffingwell, Dean, 2020), which give it a similar meaning. Also,

the team can adjust and refine the WIP limits of their task board to improve and optimize flow (Leffingwell, Dean, 2020).

4.4.5 Task management

The AGSD practice of task management focuses on assigning tasks to the team members based on their skills and roles, and to make them visible to anyone in the distributed sites (KAUSAR; AL-YASIRI, 2017). This practice can be linked to the development process visibility from SAFe (Leffingwell, Dean, 2020). SAFe practice aims to make development process visible to anyone to help all team members to visualize the stories and their progress throughout the iteration (Leffingwell, Dean, 2020).

4.4.6 Agile coaching

Agile coaching practice from AGSD studies focuses on training the global teams in the agile process (LEE; YONG, 2009) and specifically in coding skills (S.; KUMAR; MANI, 2018). Such a practice can be linked to cross-training from SAFe (Leffingwell, Dean, 2020), which aims to make the team members develop skills in new domains, development languages, and systems. Both AGSD and SAFe have specific practices to coaching the teams, although in AGSD coaching in the agile process was more explicit (LEE; YONG, 2009).

4.4.7 Document necessary information

This practice can be linked to two practices from SAFe: readiness activities (Leffingwell, Dean, 2020), and vision presentation continuously (Leffingwell, Dean, 2020). It aims on documenting and making available the necessary information about the project that generates value both to the customer and to global teams (AVRITZER; BRONSARD; MATOS, 2010; Nuevo; Piattini; Pino, 2011). From the other side, SAFe readiness activities usually happen in innovation and planning iterations, and documentation happens at this moment because it is not feasible or economical to perform in every iteration (Leffingwell, Dean, 2020). Besides it, SAFe has the practice of vision presentation continuously, which consists of replacing the vision documentation by rolling-wave vision briefings (Leffingwell, Dean, 2020). Instead of heavy documentation, this practice provides periodically presentations of the short and long terms vision to the teams (Leffingwell, Dean, 2020).

4.4.8 Synchronize work hours

In an AGSD, this practice means to establish a common time frame for all global teams from different time zones, whenever it is possible (HILLEGERSBERG; LIGTEN-

BERG; AYDIN, 2011; BASS, 2015; Nuevo; Piattini; Pino, 2011). Such practice intends to increase communication, collaboration among teams, and reduce misunderstandings. It can be linked to the practice of common cadence and synchronization from SAFe (Leffingwell, Dean, 2020), that establishes the use of a regular and predictive development rhythm (Leffingwell, Dean, 2020). Both AGSD and SAFe practice can be linked since they have the goal to establish common times for the teams to have synchronous communication and hold meetings.

4.4.9 Project wiki

The building of a project wiki was seen in some AGSD studies, and it aims on concentrating all the information regarding the global project and making it available to anyone at different sites (AVRITZER; BRONSARD; MATOS, 2010; LEE; YONG, 2009; Nuevo; Piattini; Pino, 2011). Similarly, SAFe suggests the use of a wiki to provide alignment for remote team members by structured sources of information, besides information regarding strategic themes, working rules, agreements, information about the solution, etc (Leffingwell, Dean, 2020).

4.4.10 System demo

The system demo of AGSD studies has the same process and goals of the SAFe system demo. It consists of the team showing the current state of the solution to the product owner, stakeholders, and customers, intending to obtain feedback about the work done (DINGSøYR et al., 2017). At SAFe is not different, the teams would demo their work to the product owner as soon as they are done, allowing teams to get their work completed and ready to show by the end of the iteration (Leffingwell, Dean, 2020).

4.4.11 Assign a role to each project member

Both AGSD and SAFe aims to form a cross-functional team capable of executing all tasks needed to achieve the solution goals. Assign roles to each project member is a practice that intends to assign the team members roles based on their skills (Gupta; Manikreddy, 2015). In AGSD some agile team roles were fully described (Gupta; Manikreddy, 2015; LI; MäDCHE, 2013), and such practice can be linked to Agile teams (Leffingwell, Dean, 2020) and leadership roles (Leffingwell, Dean, 2020) from SAFe. The basic Agile teams from SAFe have a product owner, scrum master role, and the team (Leffingwell, Dean, 2020). Furthermore, the practice of leadership roles can also be linked since it consists of letting solution architect/engineering become responsible for teaching, mentoring, and helping the effectiveness of Agile teams (Leffingwell, Dean, 2020).

4.4.12 Collaboration among teams

AGSD studies and SAFe share the same goal to reinforce the relationships among teams. It aims to increase collaboration, strengthen the bonds of team members, and generate trust (Paasivaara, 2017; DINGSøYR et al., 2017). The collaboration practice from AGSD can be combined with many SAFe practices related to collaboration. Agile teams are encouraged to collaborate with the PO to create, refine, and define user stories and acceptance tests for them(Leffingwell, Dean, 2020). The collaboration and organization practice intends to promote collaboration among agile teams and IT organizations to ensure the DevOps process is working well and the solutions are being developed and delivered in a reliable way (Leffingwell, Dean, 2020). The collaboration between solution and system architects aims to ensure that solutions created by each ART and suppliers fit the larger capabilities and are in the direction of the overall solution (Leffingwell, Dean, 2020). The features definition practice aims to suggest that product managers, product owners, and stakeholders work collaboratively to define features in ART's local context, mostly based on epics (Leffingwell, Dean, 2020). Finally, agile teams and business owners' proximity aims to promote face-to-face dialogue between them and a better understanding of business objectives, stimulate personal relationships bonds, and assign business values to feature (Leffingwell, Dean, 2020).

4.4.13 Agile architecture

Agile architecture aims to guarantee the solution architecture design while user needs are being developed (HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; Jha; Vilardell; Narayan, 2016). In SAFe agile architecture combines the DevOps mindset, and allows the solution architecture to evolve continuously over time, while the user's current needs are being supported (Leffingwell, Dean, 2020). Furthermore, SAFe has the architecture runaway that is one of the primary tools to support the agile architecture that supports the implementation of near-term features without the need for a redesign (Leffingwell, Dean, 2020). SAFe also describes the architecture incrementally practice that suggests the enterprise to implement architecture features incrementally by individual ARTs' through a program increment (Leffingwell, Dean, 2020). Besides it, the intentional architecture practice provides the guidance needed for inter-team design, and to ensure the system is fit for its purpose (Leffingwell, Dean, 2020). Finally, the modular architecture practice consists of standards interfaces among components to allow smaller component changes to be published with fewer risk (Leffingwell, Dean, 2020).

4.4.14 Coding standards

Coding standards aim to formalize common standards for writing readable and sustainable code that should be followed by all developers in the project (MARUPING, 2010; LEE; YONG, 2009). SAFe coding standards have similar purposes, it encourages code consistency to make everyone able to understand and maintain the quality of each solution component (Leffingwell, Dean, 2020). SAFe also suggests the use of design patterns to provide a common language to ease understand and readability (Leffingwell, Dean, 2020). SAFe suggests the use of SOLID principles too (Leffingwell, Dean, 2020) to make implementations easier to understand and modify, and consequently let the solution more flexible and able to support new requirements. Finally, coding standards can be extended to modular architecture from SAFe which aims to build a software architecture with standard interfaces among components to allow small component changes to be released independently with fewer risks (Leffingwell, Dean, 2020).

4.4.15 Share mission and vision

This practice aligns all global teams to the same vision and mission for implementing the solution based on feasible requirements that attend customer needs (Gupta; Manikreddy, 2015; LEE; YONG, 2009). In SAFe, it consists on the team planning together in the PI planning to align a shared mission and vision (Leffingwell, Dean, 2020). Also, SAFe has the practice of a holistic and cohesive vision in which product and solution management work with business owners and stakeholders to synthesize all the inputs from them and integrate it into a holistic and cohesive vision (Leffingwell, Dean, 2020). Furthermore, SAFe has vision presentation continuously that replaces vision documentation for vision briefings through short presentations about short- and long-terms of the vision to the teams (Leffingwell, Dean, 2020). Finally, SAFe has the presentations of the top ten features for the next PI. It helps the team to better understand the vision and points that need to be developed urgently (Leffingwell, Dean, 2020).

4.4.16 Simple design

AGSD simple design aims to keep the solution design and architecture as simple as possible along the development process (LEE; YONG, 2009). The practice can be linked to some specific practices of SAFe about design such as quality of architecture and design. Both practices ensure that future requirements would be easier to implement, the system would be easier to test, and non-functional requirements would be easily satisfied. Besides, the set-based design (SBD) (Leffingwell, Dean, 2020) that evaluates multiple design alternatives during the development process that aims to explore, analyze, and validate multiple choices to remove the unnecessary ones and to

make the best decision by converting uncertainties to knowledge (Leffingwell, Dean, 2020). Also, SAFe has the practice of using design patterns to provide common standards to ease understanding and readability (Leffingwell, Dean, 2020). Furthermore, the design choices consist of system architect/engineering working with the agile teams to ensure that design choices are being made with an understanding of the overall solution to minimize technology complexity and unnecessary duplication of code (Leffingwell, Dean, 2020). Finally, the emergent design (Leffingwell, Dean, 2020) practice from SAFe provides a technical basis for an incremental development process in the solution. It allows the solution design to enhance while the solution is being built and released, and consequently make the agile teams able to respond quickly to any user needs (Leffingwell, Dean, 2020).

4.4.17 Roadmap planning

The AGSD roadmap provides the customers and suppliers information to understand and collaboratively plan for the future milestones and deliverables of the solution (Leffingwell, Dean, 2020; Nuevo; Piattini; Pino, 2011). The SAFe roadmap (Leffingwell, Dean, 2020) is a schedule of events that provide all stakeholders a view of the current state of the solution, and the near- and long- terms of the deliverables of the solution. Also, continuous exploration practice consists of constantly exploring the market and user needs, defining solution vision, roadmap, and hypotheses to address those needs (Leffingwell, Dean, 2020).

4.4.18 Expand teams responsibility gradually

In AGSD, this practice consists in making the team members more independent and capable to solve problems by themselves (Nuevo; Piattini; Pino, 2011). In SAFe, the practice of stop-the-line-mentality (Leffingwell, Dean, 2020) encourages everyone to feel able to fix any problem until it's resolved (Leffingwell, Dean, 2020). It can be linked to this AGSD practice. Even if the problem is present in the deployed solution or in the continuous delivery pipeline, the stop-the-line-mentality should be present. It aims to transform the problems found in improvements and avoid it to happen again in the future (Leffingwell, Dean, 2020).

4.5 Discussion and conclusion of scaling agile practices in AGSD

"Agilists" such as Ken Schwaber, says that "SAFe is based on RUP, not Scrum" (Schwaber, Ken, 2013). Stephen Denning (board of Scrum Alliance), affirms that SAFe reinforces a return to the unproductive vertical world of hierarchical bureaucracy (DENNING, 2015). Pancholi e Grover (PANCHOLI; GROVER, 2014) argue that SAFe "kills

the spirit of agile development". However, agile large enterprises with dozens or even hundreds of global teams, complex decision-making mechanisms, often need a structured approach to implement such a significant change (Paasivaara, 2017). Furthemore, despite the statements of some authors who evaluate to what extent SAFe is truly agile, according to the Version One 2020 survey (VersionOne, Inc., 2020), SAFe is currently the most used scaled agile approach followed by scrum-of-scrums method.

Interestingly, only 19 studies from 76 AGSD studies pointed out to scale agile practices. Although, 57 AGSD studies are related to medium and large projects, but without giving proper attention to scaling agile. Moreover, these projects could have benefited from this approach. Most of these studies adapted regular scrum practices to global scenarios, such as sprints, daily meetings, sprint reviews, continuous delivery, etc (BJØRN; SØDERBERG; KRISHNA, 2019; Sundararajan; Bhasi; Vijayaraghavan, 2014; HOLE; MOE, 2008). However, they complained about the difficulty to synchronize work hours, align all members (BJØRN; SØDERBERG; KRISHNA, 2019), deal with hierarchical structures (Sundararajan; Bhasi; Vijayaraghavan, 2014), and maintain constant communication among teams (HOLE; MOE, 2008).

The literature on AGSD present alternatives for these problems such as *Synchronize work hours* (4.4.8) to better manage the team members' work journey and align them by combining meeting hours; *document necessary information* (4.4.7) to help align team members by making information available to anyone through documentation; *agile coaching* (4.4.6) practice to manage the hierarchical structure with some IT vendors and customers; and *communication practices* (4.4.1), such as "multiple communication modes" that could help those projects to establish a communication strategy, and consequently better manage the communication among teams.

The SLR showed 18 scaling AGSD practices, that could be directed linked to Scaled Agile Framework (Leffingwell, Dean, 2020). It reinforced that SAFe is a framework aligned with what is applied in GSD and large-scale agile development projects.

Despite the evidence that AGSD's practices are aligned with SAFe, only three out of fifteen case studies adopted SAFe, as presented in table 6. It is possible that the unfamiliarity with SAFe leads AGSD companies to adopt regular and known agile methods instead of SAFe. According to the studies short releases cycles, positive cultural changes, communication and interaction reinforcement among teams, friendly approach for requirement changes, management of dependencies, continuous integration, and specific practices such as distributed pair programming (GUPTA; JAIN; SINGH, 2018; TRIPATHI et al., 2015; LI; MäDCHE, 2013; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; MARUPING, 2010) are some of the reasons reported to chose methods such as Scrum, XP and Kanban. In the meantime, the studies that used SAFe,

stated that SAFe was chosen due to its scalability and modularity to adopt agile at an enterprise level, to increase productivity, improve code quality, and also because it combines Scrum, XP and Kanban practices (RAZZAK et al., 2018; Paasivaara, 2017; RAZZAK et al., 2017).

Adopt SAFe instead of scaling regular agile methods does not prevent global teams to face many challenges in large distributed projects. Among them, *synchronize work hours* 4.4.8 with team members from different time zones is still difficult even with SAFe. Organizing multiple events to happen at the same time could also lead to a big amount of effort to later align all the teams. Furthermore, *visits among sites* 4.4.3 and the linked practice Gemba walks (Leffingwell, Dean, 2020) are not always possible to apply since it is not affordable for any project to send their teams to other sites or the customer site. Due to it, it is necessary to evaluate each scenario to better understand which SAFe practices make sense to be adopted since not all practices listed can be applied in any AGSD project.

Based on Dingsoyr taxonomy *et al.* (DINGSØYR; FÆGRI; ITKONEN, 2014), it is important to point that the selected case studies are large-scale projects or very large-scale projects and that only two studies did not report the number of teams (GUPTA; JAIN; SINGH, 2018; Nuevo; Piattini; Pino, 2011). The large-scale projects represent only four case studies, two of them used SAFe in their development process (RAZZAK et al., 2018; RAZZAK et al., 2017), and the others tailored Lean and Scrum methods to a scaled level (S.; KUMAR; MANI, 2018; Gupta; Manikreddy, 2015). The majority of large-scale projects reported were executed with global teams from Europe, North America, and India.

Eleven of the AGSD studies are pointed out as very large-scale projects. From those, just one project reported the use of SAFe (Paasivaara, 2017), most of them used scaled agile methods such as, Scrum (LI; MäDCHE, 2013; HILLEGERSBERG; LIGTENBERG; AYDIN, 2011; AVRITZER; BRONSARD; MATOS, 2010; LEE; YONG, 2009; Jha; Vilardell; Narayan, 2016), XP (MARUPING, 2010) and Kanban (TRIPATHI et al., 2015), finally, one study presented a mixed framework that combined XP and Scrum practices (Nuevo; Piattini; Pino, 2011). Furthermore, it is possible to say that Scrum and its adaptations are the most chosen framework for AGSD in large projects. Although, to make Scrum safe by scaling up, sometimes we need to make it SAFe.

At last, the contrast among the case studies was compelling, with projects that needed to scale agile with three teams and more than 70 contributors (RAZZAK et al., 2017) to studies with over 30 thousand contributors (Nuevo; Piattini; Pino, 2011) or even with 73 teams (MARUPING, 2010).

However, while the agile approach suggests light and thin methods, it cannot avoid dealing with high-level complexity at the company level. Even the most qualified

team needs know which direction to go to create value for the customer. This direction must come from the company's strategy. The virtue of the large-scale agile approach is not to dominate lean, Kanban, XP, or Scrum, but to maintain the two-way flow of communication between corporate strategic decisions and IT projects. SAFe combines several AGSD practices and provides a comprehensive view from the top level of the organization to the team. Some of the applied methods are adjustable and are all connected to allow strategic alignment with the execution of IT projects in a way that supports fast and frequent deliveries of a software product.

4.6 Chapter conclusion

The present chapter shows and discusses the results obtained through the application of the SLR. Besides it, we present the mapping between the scaling agile and SAFe practices.

Initially, it details the results of the systematic review of how practices are adopted in global teams. From the 76 studies selected, 48 practices were extracted and each of them was described according to the evidences present in the studies.

Moreover, after the extraction of 48 agile practices used in the GSD context, the studies that stated the use of scaling agile practices were evaluated. From 76 reviewed studies, 19 articles were selected. From those articles, 18 scaling agile practices of the 48 agile practices were identified. Furthermore, the mapping of those practices with SAFe practices were presented.

Finally, with the results of the SLR, It was possible to answer the research questions, provide evidence on how agile and scaling agile practices are applied in globally distributed environments, also present that AGSD and SAFe practices are related.

5 Final consideration

Agile global software development can be considered as a popular trend to develop software, as many companies engage to adopt agile methods to better coordinate the development process of a solution in a distributed context.

This study carried a detailed systematic review of how practices are adopted in global teams. 76 studies published between 2004 and 2019 were selected. These studies include 48 agile practices used in global software development. The literature has shown us that these practices are being adapted to the global context, mitigating the challenges of the GSD. Also, our study showed that it is possible to adopt agility in GSD projects without the support of traditional plan-oriented approaches (Szabó; Steghöfer, 2019; LOUS; KUHRMANN; TELL, 2017; LAL; CLEAR, 2018; RAMESH et al., 2006; RICHTER; RAITH; WEBER, 2016; SIEVI-KORTE; RICHARDSON; BEECHAM, 2019).

Furthermore, it was presented the mapping of 18 agile scaling practices that were found in the 19 AGSD studies from the SLR with the SAFe 5.0 (Leffingwell, Dean, 2020) practices. Also, it was shown practices of SAFe and how to implement them could contribute to the implementation of the AGSD practices. The mapping guides by showing where the SAFe practices can fit in AGSD projects. Furthermore, the linked practices can help global teams that need to scale agile in distributed environments.

The study contributes to the area of GSD, AGSD, and large-scale agile development by showing how to implement and adapt agile in GSD contexts, and also showing that large-scale agile projects are usually conducted with global teams. Also, the study help practitioners and researchers that can now see the AGSD practices are related to SAFe 5.0, and this framework could be the choice to achieve better levels of coordination in their projects.

5.1 Limitations and threats to validity

The SLR was conducted following a solid research protocol developed and discussed by the authors, although it still has some threats to validity.

Construct Validity defines in what degree the operational measures that are studied, really represent what the researchers intended to look for and what is investigated according to the research question (WOHLIN et al., 2012). To reduce this threat the advisor was involved in the extraction of agile practices and in the mapping phase. Both discussed each extracted and mapped practice, and if any of them disagreed with the link, they would discuss to reach in a consensus.

External Validity is related to what extent it is possible to generalize the findings, and to what extent the findings have value to practitioners and researchers (WOHLIN et al., 2012). The SLR was conducted through a research protocol developed and validated by the advisor. Also, It was selected the most renowned databases for agile research, such as ACM, IEEEXplore, Springer, Scopus, and Wiley. The search string used in pre-defined bibliographic databases, which are references in agile development, ensures certain generalism of the findings since the most articles of the area are published on those databases. Furthermore, another point that ensures our external validity is pointed by the fact that the research findings have been of interest to practitioners and researchers since the early 2000s (Razavi; Ahmad, 2014). Finally, the repetition of the research protocol using the same search engines can easier study reproduction.

Internal Validity is concerned about the effects of the treatments on the variables due to uncontrolled factors in the environment (WOHLIN et al., 2012). To mitigate this threat the research protocol was strictly followed through the analysis of the articles with the involvement of the advisor. Besides it, every article chose and practice extracted from the papers was discussed, and when one of them disagreed with a choice, all of them discussed to reach a common agreement. Such activity aimed to ensure internal validity and assure that all results would derive from the data.

5.2 Future work

During the execution of this research, it is possible to point some gaps and possibilities for future research that can be based on the results presents. The following possibility for future works are:

1. Continue the SLR to verify whether teams that claim to be agile are indeed agile or use a hybrid approach;
2. Evaluate more deeply the SLR results to identify how AGSD teams deal with uncertainties such as threats and opportunities;
3. Evaluate the mapping AGSD and SAFe practices in a real industrial development project through the execution of a case study;
4. Execute a survey with distributed agile teams to verify if SAFe and AGSD practices are being used at large-scale distributed projects;
5. Expand the study to evaluate the adoption of AGSD and SAFe practices to help manage uncertainties in agile large-scale projects.

Referências

ALSAHLI, A. A.; KHAN, H.; ALYAHYA, S. Agile development overcomes gsd challenges: A systematic literature review. *International Journal of Computer Science and Software Engineering*, v. 6, p. 7–18, 01 2017. Citado na página 22.

ALSAQAF, W.; DANEVA, M.; WIERINGA, R. Quality requirements in large-scale distributed agile projects – a systematic literature review. In: GRÜNBACHER, P.; PERINI, A. (Ed.). *Requirements Engineering: Foundation for Software Quality*. Cham: Springer International Publishing, 2017. p. 219–234. ISBN 978-3-319-54045-0. Citado 4 vezes nas páginas 39, 40, 41 e 42.

ALTAF, A. et al. A systematic literature review on factors impacting agile adaptation in global software development. In: *Proceedings of the 2019 7th International Conference on Computer and Communications Management*. New York, NY, USA: Association for Computing Machinery, 2019. (ICCCM 2019), p. 158–163. ISBN 9781450371957. Disponível em: <https://doi.org/10.1145/3348445.3348463>. Citado na página 15.

ALZOUBI, Y.; GILL, A.; AL-ANI, A. Empirical studies of geographically distributed agile development communication challenges: A systematic review. *Information & Management*, v. 53, 08 2015. Citado na página 22.

AMBLER, S. W.; LINES, M. *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. [S.l.]: IBM press, 2012. Citado na página 15.

AVRITZER, A.; BRONSARD, F.; MATOS, G. Improving global development using agile. In: ____. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 133–148. ISBN 978-3-642-12442-6. Disponível em: <https://doi.org/10.1007/978-3-642-12442-6_9>. Citado 20 vezes nas páginas 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 49, 51, 55, 56, 57, 58, 59, 60 e 65.

BANIJAMALI, A. et al. Empirical investigation of scrumban in global software development. In: HAMMOUDI, S. et al. (Ed.). *Model-Driven Engineering and Software Development*. Cham: Springer International Publishing, 2017. p. 229–248. ISBN 978-3-319-66302-9. Citado 2 vezes nas páginas 37 e 44.

BASS, J. How product owner teams scale agile methods to large distributed enterprises. *Empirical Software Engineering*, v. 20, p. 1525–1557, 12 2015. Citado 6 vezes nas páginas 36, 37, 43, 56, 57 e 60.

BECK, K. Extreme programming explained: Embrace change. Addison-Wesley Professional, 2000. Citado 2 vezes nas páginas 18 e 20.

BECK, K. et al. *Manifesto for Agile Software Development*. 2001. Disponível em: <http://www.agilemanifesto.org/>. Citado 5 vezes nas páginas 15, 18, 19, 20 e 54.

Betz, S.; Makio, J.; Stephan, R. Offshoring of software development - methods and tools for risk management. In: *International Conference on Global Software*

Engineering (ICGSE 2007). Munich, Germany: IEEE, 2007. p. 280–281. Citado na página 19.

BITTNER, K. et al. *The Nexus Framework for Scaling Scrum: Continuously Delivering an Integrated Product with Multiple Scrum Teams*. [S.l.]: Addison-Wesley Professional, 2017. Citado na página 15.

BJØRN, P.; SØDERBERG, A.-M.; KRISHNA, S. Translocality in global software development: the dark side of global agile. *Human–Computer Interaction*, v. 34, p. 174 – 203, 2019. Citado 2 vezes nas páginas 36 e 64.

Britto, R.; Mendes, E.; Börstler, J. An empirical investigation on effort estimation in agile global software development. In: *2015 IEEE 10th International Conference on Global Software Engineering*. Ciudad Real, Spain: IEEE, 2015. p. 38–45. Citado 2 vezes nas páginas 37 e 47.

BRITTO, R.; USMAN, M.; MENDES, E. Effort estimation in agile global software development context. In: DINGSØYR, T. et al. (Ed.). *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*. Cham: Springer International Publishing, 2014. p. 182–192. ISBN 978-3-319-14358-3. Citado 3 vezes nas páginas 37, 47 e 49.

CHO, J. Distributed scrum for large-scale and mission-critical projects. In: *AMCIS*. [S.l.: s.n.], 2007. v. 1, p. 235. Citado 3 vezes nas páginas 37, 44 e 52.

CONBOY, K. et al. People over process: key people challenges in agile development. 2011. Citado na página 18.

DENNING, S. Agile: it's time to put it to use to manage business complexity. *Strategy & Leadership*, Emerald Group Publishing Limited, 2015. Citado na página 63.

DIKERT, K.; PAASIVAARA, M.; LASSENIUS, C. Challenges and success factors for large-scale agile transformations : A systematic literature review. *Journal of Systems and Software*, v. 53, p. 87–108, 2016. Citado na página 23.

DIKERT, K.; PAASIVAARA, M.; LASSENIUS, C. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, Elsevier, v. 119, p. 87–108, 2016. Citado 2 vezes nas páginas 15 e 16.

DINGSØYR, T.; FÆGRI, T. E.; ITKONEN, J. What is large in large-scale? a taxonomy of scale for agile software development. In: *PROFES*. [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 55 e 65.

DINGSøYR, T. et al. Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering*, v. 23, p. 1–31, 06 2017. Citado 16 vezes nas páginas 38, 39, 40, 45, 46, 47, 48, 49, 50, 53, 55, 56, 57, 58, 60 e 61.

DORAIRAJ, S.; NOBLE, J.; MALIK, P. Bridging cultural differences: A grounded theory perspective. In: *Proceedings of the 4th India Software Engineering Conference*. New York, NY, USA: Association for Computing Machinery, 2011. (ISEC '11), p. 3–10. ISBN 9781450305594. Disponível em: <https://doi.org/10.1145/1953355.1953357>. Citado 2 vezes nas páginas 48 e 49.

DORAIRAJ, S.; NOBLE, J.; MALIK, P. Understanding team dynamics in distributed agile software development. In: WOHLIN, C. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 47–61. Citado 9 vezes nas páginas 18, 36, 37, 38, 39, 46, 48, 49 e 52.

DUMITRIU, F.; OPREA, D.; MESNITA, G. Issues and strategy for agile global software development adoption. *Proceedings of the World Multiconference on Applied Economics, Business and Development*, v. 209, p. 37–42, 01 2011. Citado 3 vezes nas páginas 37, 38 e 44.

DYBA, T.; DINGSOYR, T.; HANSSEN, G. K. Applying systematic reviews to diverse study types: An experience report. In: *1st Int'l Conference on Empirical Software Engineering and Measurement (ESEM) 2007.* Madrid, Spain: IEEE, 2007. p. 225–234. Citado na página 29.

ESTáCIO, B.; PRIKLADNICKI, R. A set of practices for distributed pair programming. In: *ICEIS*. [S.l.: s.n.], 2014. v. 2. Citado 4 vezes nas páginas 36, 39, 41 e 44.

Gervigny, M. L. I.; Nagowah, S. D. Knowledge sharing for agile distributed teams: A case study of mauritius. In: *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*. Dubai, United Arab Emirates: IEEE, 2017. p. 413–419. Citado na página 42.

GONÇALVES, W. F. et al. Using agile methods in distributed software development environments. In: SILVA, T. Silva da et al. (Ed.). *Agile Methods*. Cham: Springer International Publishing, 2017. p. 16–27. ISBN 978-3-319-55907-0. Citado na página 47.

GUPTA, R. K.; JAIN, S.; SINGH, B. Challenges in scaling up a globally distributed legacy product: A case study of a matrix organization. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 77–81. ISBN 9781450357173. Disponível em: <https://doi.org/10.1145/3196369.3196389>. Citado 9 vezes nas páginas 38, 40, 42, 45, 55, 56, 57, 64 e 65.

Gupta, R. K.; Manikreddy, P. Challenges in adapting scrum in legacy global configurator project. In: *2015 IEEE 10th International Conference on Global Software Engineering*. Ciudad Real, Spain: IEEE, 2015. p. 46–50. Citado 17 vezes nas páginas 36, 37, 38, 39, 40, 41, 42, 50, 51, 52, 55, 56, 57, 58, 60, 62 e 65.

Hamid, A. M. E. Upgrading distributed agile development. In: *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*. Khartoum, Sudan: IEEE, 2013. p. 709–714. Citado 6 vezes nas páginas 36, 37, 40, 44, 45 e 48.

HILLEGERSBERG, J. van; LIGTENBERG, G.; AYDIN, M. N. Getting agile methods to work for cordys global software product development. In: KOTLARSKY, J.; WILLCOCKS, L. P.; OSHRI, I. (Ed.). *New Studies in Global IT and Business Service Outsourcing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 133–152. ISBN 978-3-642-24815-3. Citado 19 vezes nas páginas 15, 18, 36, 39, 40, 41, 42, 43, 45, 48, 50, 55, 56, 57, 58, 60, 61, 64 e 65.

HOLE, S.; MOE, N. B. A case study of coordination in distributed agile software development. In: O'CONNOR, R. V. et al. (Ed.). *Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 189–200. ISBN 978-3-540-85936-9. Citado 12 vezes nas páginas 18, 36, 37, 38, 39, 40, 42, 43, 47, 48, 49 e 64.

Hossain, E.; Babar, M. A.; Paik, H. Using scrum in global software development: A systematic literature review. In: *2009 Fourth IEEE International Conference on Global Software Engineering*. Limerick, Ireland: IEEE, 2009. p. 175–184. Citado 18 vezes nas páginas 19, 21, 36, 37, 38, 39, 41, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53 e 55.

HOSSAIN, E.; BABAR, M. A.; VERNER, J. How can agile practices minimize global software development co-ordination risks? In: O'CONNOR, R. V. et al. (Ed.). *Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 81–92. ISBN 978-3-642-04133-4. Citado 8 vezes nas páginas 36, 37, 39, 42, 46, 47, 50 e 51.

HOSSAIN, E.; BABAR, M. A.; VERNER, J. Towards a framework for using agile approaches in global software development. In: BOMARIUS, F. et al. (Ed.). *Product-Focused Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 126–140. ISBN 978-3-642-02152-7. Citado 8 vezes nas páginas 36, 38, 43, 44, 45, 47, 48 e 52.

HOSSAIN, E.; BANNERMAN, P. L.; JEFFERY, D. R. Scrum practices in global software development: A research framework. In: CAIVANO, D. et al. (Ed.). *Product-Focused Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 88–102. ISBN 978-3-642-21843-9. Citado 8 vezes nas páginas 36, 37, 38, 39, 40, 42, 43 e 46.

HOSSAIN, E.; BANNERMAN, P. L.; JEFFERY, R. Towards an understanding of tailoring scrum in global software development: A multi-case study. In: *Proceedings of the 2011 International Conference on Software and Systems Process*. New York, NY, USA: Association for Computing Machinery, 2011. (ICSSP '11), p. 110–119. ISBN 9781450307307. Disponível em: <https://doi.org/10.1145/1987875.1987894>. Citado 9 vezes nas páginas 36, 37, 38, 39, 40, 41, 42, 43 e 44.

HOSSAIN, S. S. Challenges and mitigation strategies in reusing requirements in large-scale distributed agile software development: A survey result. In: ARAI, K.; BHATIA, R.; KAPOOR, S. (Ed.). *Intelligent Computing*. Cham: Springer International Publishing, 2019. p. 920–935. Citado 3 vezes nas páginas 37, 38 e 41.

HUBER, T.; DIBBERN, J. How collaboration software enables globally distributed software development teams to become agile - an effective use perspective. In: KOTLARSKY, J.; OSHRI, I.; WILLCOCKS, L. P. (Ed.). *Governing Sourcing Relationships. A Collection of Studies at the Country, Sector and Firm Level*. Cham: Springer International Publishing, 2014. p. 49–63. ISBN 978-3-319-11367-8. Citado 4 vezes nas páginas 36, 37, 42 e 45.

Jalali, S.; Wohlin, C. Agile practices in global software engineering - a systematic map. In: *2010 5th IEEE International Conference on Global Software Engineering*. Princeton, NJ, USA: IEEE, 2010. p. 45–54. Citado 15 vezes nas páginas 18, 36, 37, 39, 40, 41, 42, 43, 45, 46, 48, 49, 50, 51 e 53.

JALALI, S.; WOHLIN, C. Global software engineering and agile practices: A systematic review. *Journal of Software: Evolution and Process*, v. 24, 10 2012. Citado 2 vezes nas páginas 19 e 21.

Jha, M. M.; Vilardell, R. M. F.; Narayan, J. Scaling agile scrum software development: Providing agility and quality to platform development by reducing time to market. In: *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. Irvine, CA, USA: IEEE, 2016. p. 84–88. Citado 7 vezes nas páginas 50, 54, 55, 56, 57, 61 e 65.

KAUSAR, M.; AL-YASIRI, A. Using distributed agile patterns for supporting the requirements engineering process. In: ____. *Requirements Engineering for Service and Cloud Computing*. Cham: Springer International Publishing, 2017. p. 291–316. ISBN 978-3-319-51310-2. Disponível em: <https://doi.org/10.1007/978-3-319-51310-2_13>. Citado 14 vezes nas páginas 36, 37, 38, 39, 40, 41, 42, 43, 44, 49, 56, 57, 58 e 59.

Khmelevsky, Y.; Li, X.; Madnick, S. Software development using agile and scrum in distributed teams. In: *2017 Annual IEEE International Systems Conference (SysCon)*. Montreal, QC, Canada: IEEE, 2017. p. 1–4. Citado 5 vezes nas páginas 36, 37, 38, 40 e 42.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing systematic literature reviews in software engineering*. [S.l.], 2007. Citado 2 vezes nas páginas 16 e 25.

KRUCHTEN, P. Contextualizing agile software development. *Journal of software: Evolution and Process*, Wiley Online Library, v. 25, n. 4, p. 351–361, 2013. Citado na página 18.

KUSSMAUL, C.; JACK, R.; SPONSLER, B. Outsourcing and offshoring with agility: A case study. In: ZANNIER, C.; ERDOGMUS, H.; LINDSTROM, L. (Ed.). *Extreme Programming and Agile Methods - XP/Agile Universe 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 147–154. ISBN 978-3-540-27777-4. Citado 5 vezes nas páginas 36, 37, 43, 44 e 49.

LAL, R.; CLEAR, T. Enhancing product and service capability through scaling agility in a global software vendor environment. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 59–68. ISBN 9781450357173. Disponível em: <https://doi.org/10.1145/3196369.3196378>. Citado 14 vezes nas páginas 36, 37, 39, 40, 41, 42, 45, 46, 47, 48, 49, 52, 53 e 67.

LARMAN, C.; VODDE, B. *Large-scale scrum: More with LeSS*. [S.l.]: Addison-Wesley Professional, 2016. Citado na página 15.

LAUREN, B. S. Mapping the workspace of a globally distributed "agile" team. *Int. J. Sociotechnology Knowl. Dev.*, IGI Global, USA, v. 7, n. 2, p. 45–62, abr. 2015. ISSN 1941-6253. Disponível em: <https://doi.org/10.4018/IJSKD.2015040104>. Citado na página 37.

LEE, S.; YONG, H.-S. Distributed agile: project management in a global environment. *Empirical Software Engineering*, v. 15, p. 204–217, 2009. Citado 20 vezes nas páginas 36, 37, 38, 40, 42, 43, 44, 46, 47, 50, 51, 52, 53, 55, 56, 57, 59, 60, 62 e 65.

LEFFINGWELL, D. *Scaling software agility: best practices for large enterprises*. [S.l.]: Pearson Education, 2007. Citado na página 15.

Leffingwell, Dean. *Scaled Agile Framework®*. 2020. <https://www.scaledagileframework.com/>. [Online; accessed 16-July-2020]. Citado 15 vezes nas páginas 15, 16, 20, 21, 25, 32, 58, 59, 60, 61, 62, 63, 64, 65 e 67.

LI, Y.; MäDCHE, A. Formulating effective coordination strategies in agile global software development teams. In: *Digital Innovation in the Service Economy. 33th International Conference on Information Systems (ICIS), Orlando, USA, December 16-19, 2012. Vol.: 5*. Orlando, FL, USA: Red Hook, Curran (NY), 2013. p. 4438–4449. ISBN 978-1-62748-604-0. Citado 10 vezes nas páginas 38, 40, 45, 50, 55, 56, 57, 60, 64 e 65.

LOUS, P.; KUHRMANN, M.; TELL, P. Is scrum fit for global software engineering? In: *Proceedings of the 12th International Conference on Global Software Engineering*. Buenos Aires, Argentina: IEEE Press, 2017. (ICGSE '17), p. 1–10. ISBN 9781538615874. Disponível em: <https://doi.org/10.1109/ICGSE.2017.13>. Citado 10 vezes nas páginas 36, 37, 38, 39, 40, 41, 42, 43, 48 e 67.

LOUS, P. et al. From scrum to agile: A journey to tackle the challenges of distributed development in an agile team. In: *Proceedings of the 2018 International Conference on Software and System Process*. New York, NY, USA: Association for Computing Machinery, 2018. (ICSSP '18), p. 11–20. ISBN 9781450364591. Disponível em: <https://doi.org/10.1145/3202710.3203149>. Citado 3 vezes nas páginas 36, 41 e 44.

LOUS, P. et al. Virtual by design: How a work environment can support agile distributed software development. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 102–111. ISBN 9781450357173. Disponível em: <https://doi.org/10.1145/3196369.3196374>. Citado 2 vezes nas páginas 15 e 36.

Marinho, M.; Noll, J.; Beecham, S. Uncertainty management for global software development teams. In: *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*. Coimbra, Portugal: IEEE, 2018. p. 238–246. Citado na página 19.

MARINHO, M. L. et al. Plan-driven approaches are alive and kicking in agile global software development. In: *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Porto de Galinhas, Brazil: IEEE, 2019. p. 1–11. Citado na página 20.

MARUPING, L. M. Implementing extreme programming in distributed software project teams: Strategies and challenges. In: ____. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 11–30. ISBN 978-3-642-12442-6. Disponível em: <https://doi.org/10.1007/978-3-642-12442-6_2>. Citado 14 vezes nas páginas 18, 37, 40, 41, 45, 49, 50, 51, 55, 56, 57, 62, 64 e 65.

MODI, S.; ABBOTT, P.; COUNSELL, S. Exploring the emergence of collaborative practices in globally distributed agile software development. In: *AMCIS*. Boston, USA: Association for Information Systems, 2017. p. 10. Citado 5 vezes nas páginas 15, 36, 41, 42 e 44.

MOE, N. et al. Coaching a global agile virtual team. In: *Proceedings of the 2015 IEEE 10th International Conference on Global Software Engineering*. Ciudad Real, Spain: IEEE, 2015. Citado 11 vezes nas páginas 36, 37, 38, 39, 40, 42, 44, 48, 49, 51 e 54.

Nuevo, E. d.; Piattini, M.; Pino, F. J. Scrum-based methodology for distributed software development. In: *2011 IEEE Sixth International Conference on Global Software Engineering*. Helsinki, Finland: IEEE, 2011. p. 66–74. Citado 19 vezes nas páginas 37, 38, 40, 42, 43, 44, 45, 46, 47, 53, 54, 55, 56, 57, 58, 59, 60, 63 e 65.

PAASIVAARA, M. Adopting safe to scale agile in a globally distributed organization. In: IEEE. *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*. [S.l.], 2017. p. 36–40. Citado 2 vezes nas páginas 15 e 16.

Paasivaara, M. Adopting safe to scale agile in a globally distributed organization. In: *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*. Buenos Aires, Argentina: IEEE, 2017. p. 36–40. Citado 9 vezes nas páginas 20, 38, 49, 55, 56, 57, 61, 64 e 65.

PAASIVAARA, M.; DURASIEWICZ, S.; LASSENIUS, C. Using scrum in a globally distributed project: A case study. *Softw. Process*, John Wiley & Sons, Inc., USA, v. 13, n. 6, p. 527–544, nov. 2008. ISSN 1077-4866. Citado 13 vezes nas páginas 36, 37, 38, 39, 40, 43, 44, 45, 47, 48, 49, 52 e 53.

PAASIVAARA, M.; LASSENIUS, C. Using scrum practices in gsd projects. In: ____. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 259–278. ISBN 978-3-642-12442-6. Disponível em: <https://doi.org/10.1007/978-3-642-12442-6_17>. Citado 7 vezes nas páginas 19, 36, 37, 38, 39, 40 e 46.

PAASIVAARA, M.; LASSENIUS, C. Scaling scrum in a large distributed project. In: IEEE. *2011 International Symposium on Empirical Software Engineering and Measurement*. [S.l.], 2011. p. 363–367. Citado na página 16.

PANCHOLI, A.; GROVER, S. Scaled agile framework: a blight. *International Journal of Innovative Research and Development (IJIRD)*, Citeseer, v. 3, n. 5, p. 425–427, 2014. Citado na página 63.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. Swindon, GBR: BCS Learning & Development Ltd., 2008. (EASE'08), p. 68–77. Citado 2 vezes nas páginas 30 e 34.

POPPENDIECK, M.; POPPENDIECK, T. Lean software development: An agile toolkit. Addison-Wesley, 2003. Citado na página 20.

POPPENDIECK, M.; POPPENDIECK, T. *Implementing lean software development: From concept to cash*. [S.l.]: Pearson Education, 2007. Citado na página 18.

RAJPAL, M. Effective distributed pair programming. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 6–10. ISBN 9781450357173. Disponível em: <https://doi.org/10.1145/3196369.3196388>. Citado 3 vezes nas páginas 20, 38 e 49.

RALPH, P.; SHPORTUN, P. Scrum abandonment in distributed teams: A revelatory case. In: *Proceedings - Pacific Asia Conference on Information Systems, PACIS 2013*. Jeju Island, Korea: AIS, 2013. v. 101, p. 1–16. ISBN 9788995217016. Citado 7 vezes nas páginas 36, 38, 39, 40, 43, 46 e 50.

RAMESH, B. et al. Can distributed software development be agile? *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 49, n. 10, p. 41–46, out. 2006. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/1164394.1164418>. Citado 7 vezes nas páginas 19, 37, 40, 41, 42, 49 e 67.

RAMESH, B.; MOHAN, K.; CAO, L. Ambidexterity in agile distributed development: An empirical investigation. *Info. Sys. Research*, INFORMS, Linthicum, MD, USA, v. 23, n. 2, p. 323–339, jun. 2012. ISSN 1526-5536. Disponível em: <https://doi.org/10.1287/isre.1110.0351>. Citado 2 vezes nas páginas 39 e 49.

Razavi, A. M.; Ahmad, R. Agile development in large and distributed environments: A systematic literature review on organizational, managerial and cultural aspects. In: *2014 8th. Malaysian Software Engineering Conference (MySEC)*. Langkawi, Malaysia: IEEE, 2014. p. 216–221. Citado 4 vezes nas páginas 23, 37, 57 e 68.

RAZZAK, M. A. Transition from plan-driven to agile: An action research. In: SPRINGER. *International Conference on Product-Focused Software ProcWohlin2012ess Improvement*. [S.l.], 2016. p. 746–750. Citado na página 15.

RAZZAK, M. A. Transition from plan-driven to agile: An action research. In: ABRAHAMSSON, P. et al. (Ed.). *Product-Focused Software Process Improvement*. Cham: Springer International Publishing, 2016. p. 746–750. Citado na página 42.

RAZZAK, M. A. et al. Transition from plan driven to safe®: Periodic team self-assessment. In: FELDERER, M. et al. (Ed.). *Product-Focused Software Process Improvement*. Cham: Springer International Publishing, 2017. p. 573–585. ISBN 978-3-319-69926-4. Citado 10 vezes nas páginas 19, 23, 36, 37, 41, 42, 48, 55, 56 e 65.

RAZZAK, M. A. et al. Scaling agile across the global organization: An early stage industrial safe self-assessment. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 121–130. ISBN 9781450357173. Disponível em: <https://doi.org/10.1145/3196369.3196373>. Citado 7 vezes nas páginas 20, 36, 37, 55, 56, 57 e 65.

RICHTER, I.; RAITH, F.; WEBER, M. Problems in agile global software engineering projects especially within traditionally organised corporations: [an exploratory semi-structured interview study]. In: *Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2016. (C3S2E '16), p. 33–43. ISBN 9781450340755. Disponível em: <https://doi.org/10.1145/2948992.2949019>. Citado 6 vezes nas páginas 37, 40, 43, 44, 45 e 67.

RIZVI, B.; BAGHERI, E.; GASEVIC, D. A systematic review of distributed agile software engineering. *Journal of Software: Evolution and Process*, v. 27, 06 2015. Citado 19 vezes nas páginas 19, 22, 36, 37, 38, 39, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52 e 55.

Robinson, P. T. Communication network in an agile distributed software development team. In: *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*. Montreal, QC, Canada, Canada: IEEE, 2019. p. 100–104. Citado 3 vezes nas páginas 36, 37 e 39.

S., R. M.; KUMAR, R.; MANI, V. S. Transitioning from plan-driven to lean in a global software engineering organization: A practice-centric view. In: *Proceedings of the 13th International Conference on Global Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2018. (ICGSE '18), p. 1–5. ISBN 9781450357173. Disponível em: <https://doi.org/10.1145/3196369.3196395>. Citado 12 vezes nas páginas 36, 38, 39, 40, 41, 44, 55, 56, 57, 58, 59 e 65.

SCHENK, J.; PRECHELT, L.; SALINGER, S. Distributed-pair programming can work well and is not just distributed pair-programming. In: *Companion Proceedings of the 36th International Conference on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2014. (ICSE Companion 2014), p. 74–83. ISBN 9781450327688. Disponível em: <https://doi.org/10.1145/2591062.2591188>. Citado na página 41.

SCHWABER, K.; BEEDLE, M. *Agile Software Development with Scrum*. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN 0130676349. Citado 2 vezes nas páginas 18 e 20.

Schwaber, Ken. *UnSAFe at any speed*. 2013. <https://kenschwaber.wordpress.com/2013/08/06/unsafe-at-any-speed/>. Online; accessed 15-June-2020. Citado 2 vezes nas páginas 16 e 63.

SIEVI-KORTE, O.; RICHARDSON, I.; BEECHAM, S. Software architecture design in global software development: An empirical study. *Journal of Systems and Software*, v. 158, p. 110400, 08 2019. Citado na página 67.

Sriram, R.; Mathew, S. K. Global software development using agile methodologies: A review of literature. In: *2012 IEEE International Conference on Management of Innovation Technology (ICMIT)*. Sanur Bali, Indonesia: IEEE, 2012. p. 389–393. Citado 5 vezes nas páginas 18, 36, 39, 42 e 43.

Sundararajan, S.; Bhasi, M.; Vijayaraghavan, P. K. Case study on risk management practice in large offshore-outsourced agile software projects. *IET Software*, v. 8, n. 6, p. 245–257, 2014. Citado 15 vezes nas páginas 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 50, 52, 53 e 64.

Sutherland, Jeff and Brown, Alex. *Scrum@Sacale*. 2020. <https://www.scrumatscale.com/scrum-at-scale-guide/>. [Online; accessed 16-July-2020]. Citado na página 15.

Szabó, D. M.; Steghöfer, J. Coping strategies for temporal, geographical and sociocultural distances in agile gsd: A case study. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. Montreal, QC, Canada, Canada: IEE, 2019. p. 161–170. Citado 13 vezes nas páginas 20, 36, 37, 38, 39, 40, 41, 42, 43, 51, 53, 55 e 67.

TANNER, M.; CHIGONA, W. Towards an understanding of the contextual influences on distributed agile software development: A theory of practice perspective. *ECIS*

2012 - Proceedings of the 20th European Conference on Information Systems, v. 178, 01 2012. Citado 6 vezes nas páginas 37, 41, 42, 43, 49 e 53.

TRIPATHI, N. et al. Scaling kanban for software development in a multisite organization: Challenges and potential solutions. In: LASSENIUS, C.; DINGSØYR, T.; PAASIVAARA, M. (Ed.). *Agile Processes in Software Engineering and Extreme Programming*. Cham: Springer International Publishing, 2015. p. 178–190. Citado 13 vezes nas páginas 37, 39, 41, 42, 45, 46, 47, 51, 55, 56, 57, 64 e 65.

VALLON, R. et al. Identifying critical areas for improvement in agile multi-site co-development. *ENASE 2013 - Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering*, v. 1, p. 165–172, 01 2013. Citado 9 vezes nas páginas 36, 37, 38, 40, 41, 42, 43, 45 e 50.

VALLON, R. et al. Systematic literature review on agile practices in global software development. *Information and Software Technology*, v. 96, 12 2017. Citado 20 vezes nas páginas 15, 16, 19, 20, 21, 36, 37, 38, 39, 40, 41, 42, 43, 46, 47, 48, 49, 50, 51 e 53.

VersionOne, Inc. *14th Annual State of Agile Development Survey*. 2020. <https://stateofagile.com/#ufh-c-7027494-state-of-agile>. [Online; accessed 28-May-2020]. Citado 3 vezes nas páginas 15, 16 e 64.

WAHYUDIN, D. et al. In-time role-specific notification as formal means to balance agile practices in global software development settings. In: MEYER, B.; NAWROCKI, J. R.; WALTER, B. (Ed.). *Balancing Agility and Formalism in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 208–222. ISBN 978-3-540-85279-7. Citado na página 18.

WIERINGA, R. et al. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.*, v. 11, p. 102–107, 03 2006. Citado 2 vezes nas páginas 30 e 33.

WOHLIN, C. et al. Experimentation in software engineering. In: ____. [S.l.]: Springer, 2012. p. 123–151. ISBN 978-3-642-29043-5. Citado 2 vezes nas páginas 67 e 68.

ZAMBONI, A. B. et al. Start uma ferramenta computacional de apoio à revisão sistemática. 2010. Citado na página 27.