Agile Software Development Practices in Remote Working Contexts: A Systematic Literature Review

by

Z. Ibrahim

Thesis presented in fulltime fulfillment of the requirements for the degree of

Master of Arts in Socio-Informatics

at the Stellenbosch University

Supervisor: Dr. DA. Parry

December 2022

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2022

Abstract

Agile Software Development Practices in Remote Working Contexts: A Systematic Literature Review

Z. Ibrahim

Department of Information Science,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MA

December 2022

Agile software development in today's organisations has become increasingly remote-oriented. The accelerated adoption of global software development and enterprise social media in remote working contexts has been shown to have the potential to alleviate remote work challenges in the software industry, particularly since the inception of the Covid-19 pandemic. Media and tools pose profound implications for agile methods and practices in distributed agile software development. The objective of this study is to explore the agile practices, tools, roles, and unique challenges that describe project management in the context of global software development. Three key aspects of agile global software development are focused on: agile methods, agile practices and, the various distribution scenarios in which development occurs. Previously studies have focused on reporting the successful application of agile practices and distribution scenarios in global software development. However, less focus has been placed on the impact of the Covid-19 pandemic on agile global software development practices in general. In this study, a systematic review approach is adopted in order to update research in this domain and gather the data necessary to further understand the usage of agile methods and practices in various remote working scenarios. In this regard, the review consisted of identifying portals to search for relevant papers using the Stellenbosch University library. Through a systematic review process

these selected and studied papers provided a number of useful themes describing many aspects of agile software development in remote working contexts, relating to agile methods, practices, and the challenges thereof. Synthesizing all of the themes, the main contribution of this study to this domain is the finding that team members in global software development were faced with similar challenges when collaborating remotely with communication media. In addition to this, this study identified that Scrum-orientated practices, and Scrum methods and Scrum in combination with other agile methods remained the most frequently adopted in remote working contexts. Ultimately, given that this project represents the continuation of an ongoing research tradition in this domain, this project provides a mile-marker for the current state of agile methods in the context of global software development. Current trends are identified, explained, and compared to the recent past, with open questions framed for future investigation.

Opsomming

Agile sagteware-ontwikkelingspraktyke in afgeleë werkkontekste: 'n sistematiese literatuuroorsig

M. Z. Ibrahim

Departement Inligtingwetenskap,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid-Afrika.

Tesis: MA

Desember 2022

Agile sagteware-ontwikkeling in vandag se organisasies het al hoe meer op afstand georiënteerd geraak. Daar is getoon dat die versnelde aanvaarding van wêreldwye sagteware-ontwikkeling en sosiale media vir ondernemings in afgeleë werkkontekste die potensiaal het om uitdagings vir afgeleë werk in die sagtewarebedryf te verlig, veral sedert die ontstaan van die Covid-19-pandemie. Media en gereedskap hou diepgaande implikasies in vir ratse metodes en praktyke in verspreide ratse sagteware-ontwikkeling. Die doel van hierdie studie is om die ratse praktyke, gereedskap, rolle en unieke uitdagings wat projekbestuur beskryf in die konteks van globale sagteware-ontwikkeling te verken. Drie sleutelaspekte van ratse globale sagteware-ontwikkeling word gefokus op: ratse metodes, ratse praktyke en die verskillende verspreidingscenario's waarin ontwikkeling plaasvind. Voorheen het studies gefokus op die rapportering van die suksesvolle toepassing van ratse praktyke en verspreidingscenario's in globale sagteware-ontwikkeling. Minder fokus is egter geplaas op die impak van die Covid-19-pandemie op ratse wêreldwye sagteware-ontwikkelingspraktyke in die algemeen. In hierdie studie word 'n sistematiese oorsigbenadering aangeneem om navorsing in hierdie domein op te dateer en die data te versamel wat nodig is om die gebruik van ratse metodes en praktyke in verskeie afgeleë werkscenario's verder te verstaan. In hierdie verband het die hersiening bestaan uit die identifisering van portale om na relevante

vraestelle te soek deur die Universiteit Stellenbosch-biblioteek te gebruik. Deur 'n sistematiese oorsigproses het hierdie geselekteerde en bestudeerde referate 'n aantal nuttige temas verskaf wat baie aspekte van ratse sagteware-ontwikkeling in afgeleë werkskontekste beskryf het, met betrekking tot ratse metodes, praktyke en die uitdagings daarvan. Deur al die temas te sintetiseer, is die hoofbydrae van hierdie studie tot hierdie domein die bevinding dat spanlede in globale sagteware-ontwikkeling voor soortgelyke uitdagings te staan gekom het wanneer hulle op afstand met kommunikasiemedia saamgewerk het. Hierbenewens het hierdie studie geïdentifiseer dat Skrum-georiënteerde praktyke, en Skrummetodes en Skrum in kombinasie met ander ratse metodes die algemeenste gebly het in afgeleë werkskontekste. Uiteindelik, aangesien hierdie projek die voortsetting van 'n deurlopende navorsingstradisie in hierdie domein verteenwoordig, bied hierdie projek 'n mylmerker vir die huidige stand van ratse metodes in die konteks van globale sagteware-ontwikkeling. Huidige neigings word geïdentifiseer, verduidelik en vergelyk met die onlangse verlede, met oop vrae wat vir toekomstige ondersoek opgestel is.

Acknowledgments

I would like to acknowledge and give my warmest thanks to my supervisor (Dr. Douglas A. Parry) who made this work possible. His advice, guidance and insights helped me through all stages of this thesis. I especially thank him for challenging me and setting deadlines. I would also like to thank my family and friends for their loving support and for believing in me every step of the way. Also, special thanks to my psychologist (Dr. Maze) for improving my mental health and keeping me motivated at times of depression and anxiety during the writeup of this thesis. Lastly, I would like to extend my gratitude to my examiners, for their feedback and comments.

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

SDLC - Systems Development Lifecycle

IDD - Iterative Incremental Development

NASA - National Aeronautics and Space Administration

XP - Extreme Programming

SE - Software Engineering

GSD - Global Software Development

GSE – Global Software Engineering

CMC - Computer-Mediated Communication

ICT - Information and Communication Technology

USA - United States of America

UK – United Kingdom

MS - Microsoft

IT - Information Technology

IBM - International Business Machines Corporation

DOT - Department of Transportation

HQ - Headquarters

ESN - Enterprise Social Network

IJIM - International Journal of Information Management

ESM - Enterprise Social Media

SME - Small and Medium Enterprises

YAGNI - You Aren’t Going To Need It

LIST OF ABBREVIATIONS AND ACRONYMS

CMMI - Capability Maturity Model Integration

ROI - Return on Investment

JIT - Just in Time

TPS - Toyota Production System

VSM - Value stream mapping

ACM - Association for Computing Machinery

IEEE - Institute of Electrical and Electronics Engineers

AIS - Automatic Identification System

INSPEC - Information Service for Physics, Electronics, and Computing

DASE - Distributed Agile Software Development

FDD - Feature Driven Development

DAD - Disciplinary Agile Delivery

RQ - Research Question

PRISMA - Preferred Reporting Items for Systematic Reviews

WIP - Work In Progress

Chapter 1

Introduction

Over the preceding decade global software development (GSD) has grown substantially and is rapidly becoming the norm in many domains (Camara et al., (2020). While various definitions exist, GSD generally refers to software development, whereby members of the software development team are globally distributed across different locations (Vallon et al., 2018). The accelerated adoption of the practice was emanant in early 2020 when the Covid-19 pandemic emerged, and countries worldwide implemented lockdowns that forced people to work from home. During this time, software development companies had to adjust, and teams had to resort to remote work (Marek et al., 2021). Remote work and distributed work align closely and have similar work arrangements. More importantly, these terms often refer to the use of technologies to enable work coordination that ultimately disconnects work activities from a conventional workplace (Henry, Le Roux & Parry, 2021). Labels such as remote work, distributed work, and virtual work often embody different conceptualizations of telecommuting (Allen, Golden & Shockley, 2015). The term telework is used to connote a broader form of telecommuting that involves conducting work activities from a variety of different locations outside the central place of work (Allen, Golden & Shockley, 2015).

Successful global software organizations depend on member's coordination and interdependence, and globally distributed teams are significantly more challenging to manage compared to co-located teams (Moe et al., 2015). Some of the challenges faced by remote teams include time zones, geographical and cultural differences, language barriers, and potential lack of competency amongst members (Moe et al., (2015). Remote workers engage with informal communication like co-located teams and have greater interpersonal skills when contributing to agile practices (Deshpande et al., 2016). Moreover, informal communication can be defined as personal and interactive and is preferred in co-located teams. However, formal

interaction is suitable for agile GSD environments, but different languages can limit communication when remote workers are distributed across several locations (Alzoubi, Gill & Al-Ani, 2016).

Alongside the emergence of GSD, over the preceding two decades, software engineering has undergone its own transformation. Specifically, the field has seen dramatic evolutions in the ways in which software is produced. A key development during this time period was the emergence of Agile methods around the turn of the century. Agile methods were first formalised by a group of software developers (Agile Alliance) interested in iterative software development (Rashid & Khan, 2018). Agile methods in software development are used by self-organizing teams that focus on communication and collaboration activities that are governed by various agile practices such as sprints, standup meetings, retrospectives, and reviews which all facilitate change to add customer value (Vallon et al., 2018). In this study, agile practices (i.e., pair programming, sprint planning etc.) are defined as key components that are based on agile methods such as Scrum, XP, and Kanban, either used as its original or adapted form to deal with software development challenges.

Originally, agile methods were developed for on-site/co-located teams, however, software companies came to the realization that agile methods can support distributed teams and agile methods such as Scrum and XP and were adapted as distributed Scrum and distributed XP (Rashid & Khan, 2018). In non-co-located situations it is difficult to exercise agile methods to develop and deliver software (Winska & Dabrowski, 2020). Face-to-face interaction and coordination in co-located teams are critical for effective communication among agile teams, but relatively challenging to exercise within globally distributed environments as members are located in different parts of the world. (Alzoubi, Gill & Al-Ani, 2016).

Remote agile software development is relatively well known amongst researchers and practitioners, and many firms are moving to this type of software development (Alsahli et al., 2017). The main benefit of GSD to companies is access to international skills, markets, and flexibility of organizational needs that to some

extent reduces production, and time to market costs (Manjavacas et al., (2020). Moreover, several leading software development companies (i.e., Microsoft, IBM) are globalizing their work by outsourcing to low-cost developing countries because of the substantial reduction in project development costs (Sinha et al., 2020). However, despite the multiple benefits that GSD brings to the software industry, there are still challenges that need to be addressed. These include temporal, geographical, socio-cultural distances, knowledge, and technical challenges, among many others (Manjavacas et al., (2020).

Research on the implementation of agile practices in remote working is spread across a number of IT fields, ranging from Information Systems, Software Engineering, to Project Management. However, to date, only three key systematic reviews (i.e., Jalali & Wohlin, 2012; Rizvi et al., 2015; Vallon et al., 2018) have been conducted to synthesise extant knowledge on agile practices in remote software development teams. The purpose and contributions associated with these previous reviews vary based on their specific methods and research foci, but in general, they focus on the combination of GSD and agile practices. A full analysis of these reviews is conducted in the latter part of this thesis.

1.1 Aims and objectives

This project aims to categorise and describe the practices, tools, roles, and unique challenges that characterise agile project management in the context of global software development. The project involves developing a deep understanding of agile project management methods, remote work, and virtual teams in the context of global software development projects. After familiarisation with the research domain, specific research questions within this context were developed and defined later in this thesis.

At a high level, research questions may be addressed through either the analysis of existing case studies, the development of a new case study (using either qualitative or quantitative techniques), or at a meta-level through the analysis and synthesis of existing knowledge in this regard via, for instance, a systematic review. This project

intends to synthesise current knowledge on the implementation of agile in remote working scenarios in the context of software development teams. To this end, the investigation will concern the practices, tools, roles, and challenges currently described in the literature. More importantly, this study is a continuation of the systematic review conducted by Vallon et al., (2018) for the years 2010-2016. In order to address these objectives, extending prior work (i.e., Jalali & Wohlin, 2012; Vallon et al. 2018), this study will involve a systematic literature review of previous studies in GSD. The focus of this study is to understand the GSD field and the various agile practices used in different distribution scenarios.

1.2 Motivation for the study

With co-location held up as the 'gold standard' for agile software development, given the 'new normal' and the continued reality of remote and distributed work, there is a need to understand how agile project management practices can be implemented remotely. Although there have been reviews (i.e., Jalali & Wohlin 2012; Rizvi et al. 2015; Vallon et al. 2018) covering different aspects in the GSD domain, there is no recent study focusing on the impact of the Covid-19 pandemic on agile software development practices. Furthermore, the current state of the successful application of agile practices since the shift to remote and distributed work has not been extensively studied.

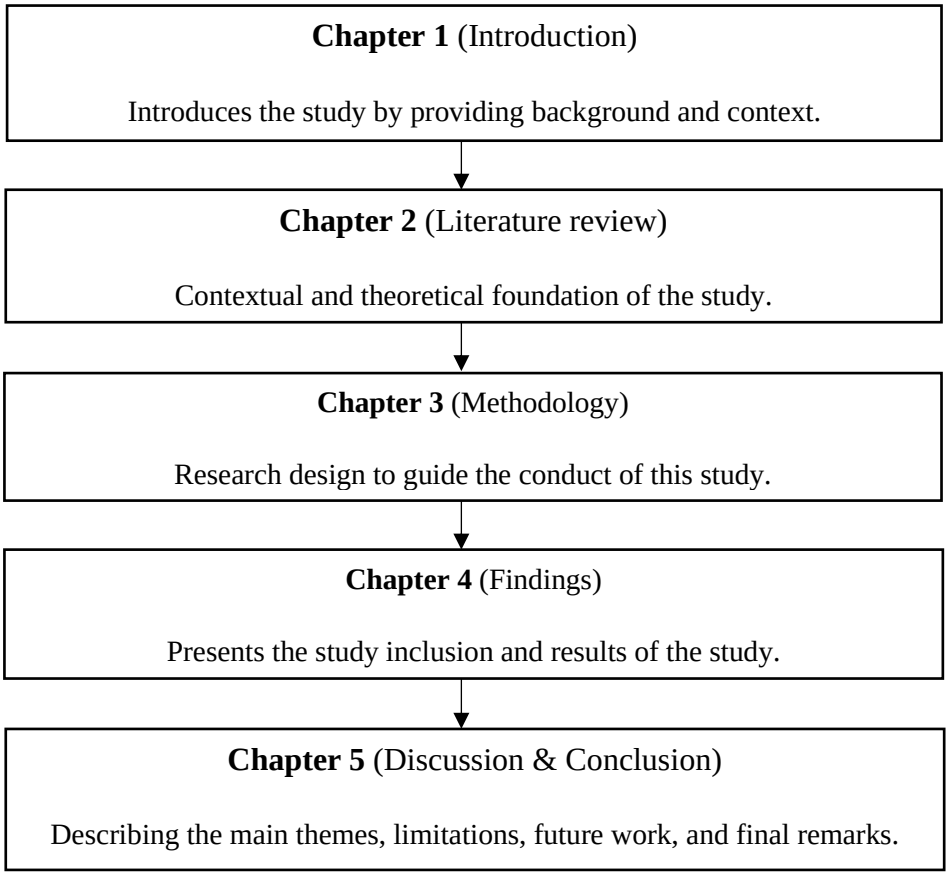
Remote work will likely continue to grow and become widespread, therefore, understanding the challenges and adoption of patterns, solutions, and methods in GSD will be useful for both researchers and software companies. According to Mahmood et al. (2022) the focus on GSD has increased because more software companies have shifted to remote work during the pandemic, and globalization has become an essential trend in today’s industry. In order to address the identified research gap, this study is built around existing studies (i.e., Jalali & Wohlin 2012; Vallon et al., 2018) and there is value in this study by continuing the work conducted by Vallon et al., (2018) to perform an updated and complete analysis, rather than covering the entire period. Global software development is an active research field and there is a need for primary studies due to the rise of remote work

and ongoing changes within this domain. To this end, this study's motivation is to answer specific research questions (see Chapter 3) in order to evaluate the current state of agile software development practices in remote working contexts. This synthesis will be useful for researchers and practitioners alike.

1.3 Outline of the thesis

The thesis is structured into five chapters (see Figure 1.1). Chapter 2 describes the literature review aspect of the thesis, whereby three bodies of literature will be discussed, namely: distributed work, agile methodologies, and extant reviews on agile practices in GSD. Following this, Chapter 3 provides details surrounding the adopted research design and research questions of the study. Chapter 4 describes the study inclusion process and reports the findings of the study. Lastly, Chapter 5 reviews all the key themes found in this study while also highlighting limitations, future directions, recommendations and concluding remarks.

Figure 1.1: Outline of the thesis



Chapter 2

Background and Literature Review

To provide a contextual background and a theoretical foundation for this study, this chapter presents a review of three relevant bodies of literature. In the first section distributed work is conceptualised with attention drawn to various remote working practices. Next, to delineate a conception of agile practices in software development contexts, the second section presents, firstly, a high-level overview of key software development methodologies, secondly, a review of the establishment of the agile movement, thirdly, a description of key characteristics of agile development and, finally, an overview of key processes in prominent agile methodologies. The third section of the chapter brings these first two elements together and presents a review of existing literature on the adoption and implementation of agile practices in remote working contexts. To this end, this third section focuses specifically on existing systematic reviews in this regard. The chapter concludes with an interpretative conclusion in which key gaps in our current knowledge on this phenomenon are summarised.

2.1 An Overview of Distributed Work Concepts

Since early 2020, the ongoing Covid-19 pandemic has led governments around the globe to enforce certain rules and regulations to minimize the spread of infection by placing restrictions on mobility. These enforcements have rapidly transformed how organizations and people conduct their work. Many central economic contributors whether they are in the public or the private sector experienced changes (e.g., a shift from central office to remote work) within their operations and employment.

To provide some background and context in this study: over the preceding 20 or so years there has been a growing trend of organisations, especially technology organisations, moving towards remote work (Brynjolfsson et al., 2020). While this

change has been slow, it has been increasing. The Covid-19 pandemic forced a dramatic increase in the proportion of workers performing their tasks remotely. This not only has short-term implications for work practices but, as many argue, it will also hold more longer-term implications. In this section, to clarify ambiguity around the definition and nature of remote work, key historical developments in the rise of remote work and important definitions for various remote working practices will be considered. Furthermore, to specifically understand team functioning as a remote working practice, the concept of a virtual team will be discussed. In addition to these concepts, given the importance of communications technologies for remote work, various digital communications technologies will be examined, with particular emphasis on how they impact and drive remote communication and collaboration.

2.1.1 Towards a Working Definition for Distributed Work

Changes in business operations enabled the continuity of economic activity during the pandemic and, whilst concepts and practices like remote work, distributed work, and virtual teams are generally well-known and established, Covid-19 has accelerated the adoption of these practices within the organizational landscape (Henry, le Roux & Parry, 2021). The location changes of daily, weekly, and monthly work patterns have altered the 'spatiality' of work: some paid work is undertaken remotely from home and other work takes place in cyberspace (Hardill & Green, 2003).

Distributed work can be defined in numerous ways, with the related concepts of telework, remote work, and virtual work frequently adopted to describe work practices in which work is performed away from some central location. It is important to note that, while these concepts differ in terms of meaning and context, they all center around the general idea of the physical or geographical distribution of work, which is "work that is not performed at a single, centralized location (Henry, le Roux & Parry, 2021, pg. 2).

The broadest concept that can be examined is telecommuting. The term telecommuting was first coined in the 1970s when practitioners and scholars debated around the idea of working away from the office as it went against traditional business operations (Allen, Golden & Shockley, 2015). Telecommuting is often used to refer to many forms of distributed work practices, and across studies, definitions of telecommuting vary. To understand an academically accepted definition, Table 2.1 provides a sample of many definitions of terms used in the literature. According to Henry, le Roux & Parry (2021, pg. 2) the selection of these definitions was achieved with a "non-random chain selection process," beginning with a seminal definition for each term. These definitions were chosen from off-cited sources. Considering these popular definitions, it is clear that the central concern of these definitions is the geographical distance in which work is conventionally conducted and the location at which telework (work being done from home or any off-site location) is performed (Henry, le Roux & Parry, 2021, pg.2).

Table 2.1: Review of Telecommuting Definitions

Concepts	Definition
Remote Work	'A work arrangement in which the employee resides and works at a location beyond the local commuting area of the employing organisation's worksite; generally, includes full-time telework and may result in a change in duty location to the alternative worksite' (US Office of Personnel Management, 2013, p. 18).
Remote and Distributed Work	'The terms remote work and distributed work are generally considered broader than telecommuting and can denote any form of work not conducted in the central office, including work at branch locations and differing business units' (Allen et al., 2015, pp. 43–44).

Distributed Work	'Employees work over geographical boundaries and to some extent work with computer-mediated communication (CMC) to achieve a common goal' (Bosch-Sijtsema & Sivunen, 2013, p. 160).
Distributed Work	'Distributed work is marked by four key features that differentiate it from more traditional office work. First, distributed work is different from more traditional office work because of the physical distance involved. Second, this physical distance is managed by a reliance on communication technology. A third feature defining distributed work is reduced supervision. Fourth, distributed work requires that individuals are interpersonally connected with some other individuals: this could be a single person, a team, or others in an organization. In other words, to be distributed means one has to be distributed from others' (Rockmann & Pratt, 2015, pg.151–152).
Telecommuting	'Telecommuting is a subset of teleworking, a similarly coined term that includes all work-related substitutions of telecommunications and related information technologies for travel' (Nilles, 1988, pg. 301).
Telework	'The term telework is generally used to connote a broader form of telecommuting that involves working from a variety of alternative locations outside of the central office (including full-time work from home but not necessarily limited to home-based work) and includes work from home-based businesses, telecentres, and call centres, and even work within an organisation's central office between individuals who are interacting using technology' (Allen et al., 2015, pg. 42–43).

Remote work and distributed work are frequently used interchangeably (O'Neill, Hambley, & Chatellier, 2014) and, considering the definitions presented in Table 2.1, it is clear that there is not a major distinction between these two concepts. However, both concepts refer to the broader definition that is – "any form of work not conducted in the central office" (Allen et al., 2015, pp. 43–44).

An additional concept used to refer to distributed work practice is that of virtual work. The concept 'virtual work' can be defined as "a broader term often used to describe individuals, groups of individuals, or organizations who do not interact face-to-face because of geographic dispersion, but who rely on using technology in some fashion to facilitate interaction" (Allen et al., 2015, p. 43). The central idea of virtual work is internet-based communication amongst distributed workers. Therefore, 'virtual' does not describe the work being virtual, but rather the virtual or simulated co-location of cooperating workers that rely on communications technologies for their communication and coordination (Henry, le Roux & Parry, 2021, pg. 4).

The definition for remote work provided by Rockman and Pratt (2015) described in Table 2.1 provides a useful conceptualization by outlining four features that characterize remote work. The first two, physical distance and use of technology overlap in distributed work. The third, reduced supervision requires consideration. It suggests that the physical location of a worker and supervisory co-workers constitutes more or increased supervision. The fourth feature states that distributed work implies communication between co-workers. The definition of remote work by Rockman and Pratt (2015) in a nutshell focuses attention on the distance between individuals, the important role of information and communication technology (ICT), the interpersonal relationships between individuals, and the different function of supervision in remote working contexts. Therefore, this definition of remote work will be adopted for the remainder of the thesis.

2.1.2 Virtual Teams as a Form of Remote Work

Virtual teamwork can have many different meanings and context across organizations. The concept of a 'team' can be described as a small number of people with complementary skills who share a common goal and are mutually accountable for their work (Ebrahim, Ahmed & Taha, 2009). Virtual teams encompass members of organizations that use information and communication technology to interact with one another across geographic boundaries of time and space. The degree of geographic dispersion within virtual teams can vary from one person located in a

different location to multiple members located across different countries, or within the same country but still geographically dispersed from each other (Ebrahim, Ahmed & Taha, 2009).

Under the conditions of Covid-19, many enterprises had to deal with challenges that hampered physical interaction and had to replace traditional contact meetings with virtual solutions. This placed enormous emphasis on the importance of virtual teams and virtual communication. This digital transformation is a requirement for team collaboration and coordination, therefore, the implementation of virtual teams within organizations had to be done swiftly and consistently after the discovery of the virus (Zeuge et al., 2020).

A recent survey conducted prior to the pandemic indicated that US corporate teams are now almost fully virtual, and 41% never meet in person and in 2016 the survey revealed that 48% of respondents confirmed that more than half of their team consisted of members from other nations.[1] The growing prevalence of virtual teams is a result of the increased adoption in communication and collaboration technologies, along with the organizational benefits associated with these virtual teams (Martins, Gilson & Maynard, 2004). With the rapid development of electronic information technology and media platforms like MS teams, Slack, Trello etc., work can be conducted faster and more efficiently (Ebrahim, Ahmed & Taha, 2009). Moreover, virtual teams enable organizations to outsource talent from far-flung places and be highly responsive to customer needs (Raghuram et al., 2019).

In defining what makes a virtual team, it is important to unpack what are some of the common virtual team characteristics found in literature. Schweitzer and Duxbury (2010), Ebrahim, Ahmed and Taha (2009) and Berry (2011) refer to frequently cited criteria, and characteristics of virtual teams:

[1]Trends in Global Virtual Teams, CultureWizard
http://cdn.culturewizard.com/PDF/Trends_in_VT_Report_4-17-2016.pdf

Common Criteria

- The members of the team may be geographically dispersed (not co-located, not working at the same location).
- Driven by a common purpose.
- The members of the team rely on computer-mediated communication rather than co-located communication to accomplish their tasks.
- Involved in cross-boundary collaboration (members from different organizational units or organizations).
- Asynchronicity (members work different times, such as different times zones or the same location, but at different time intervals).

The change from proximate, co-located work to virtual work is a process change that must be established. The effectiveness of virtual teams is likely to increase with experience and communication, hence communication in these teams needs to be unambiguous and concise (Bakshi & Krishna, 2008). The establishment of structures and virtual meetings are important to enable regular exchanges using ICTs (see Section 2.1.5). This can ultimately increase trust within the team and strengthen cooperation despite the distance (Zeuge et al., 2020). An advantage of virtual teams is that team members can communicate, collaborate, and produce work irrespective of distance and space because they are not affected by temporal constraints or geographic location like in co-located communication (Berry, 2011).

Virtual teams need to communicate and collaborate to solve problems as well as produce a product or service (Thomas, 2007). However, deciding which communication technology to use for these interactions depends on the nature and type of team, the level of access to certain technology, or even the experience and sophistication of team members and leaders within the virtual space (Berry, 2011). Shared understanding and goals are an essential component of teams to reach common ground and are an intrinsic part of the team-building process. Therefore, for effective collaborative work, virtual or co-located, requires the establishment of social relationships (Vroman & Kovachich, 2002).

The primary distinction between co-located and virtual teams is that virtual teams always use various forms of computer-mediated technology for collaboration (Berry, 2011). Virtual team interactions can be classified into one of the following four categories (Mittleman & Briggs, 1998):

- Same time and location interactions such as co-location but using ICT to facilitate communication.
- Same time but different location interactions
- Different time but same location interactions
- Different time and different location interactions, exchange of e-mail communications, and video conferencing.

These categories illustrate that most teams in some ways are virtual because co-located teams make use of electronic media for communication, therefore, both virtual and co-located teams make use of computer-mediated communication in some way or another in their interactions (Berry, 2011).

2.1.3 Global Software Development

Global software development (GSD) refers to the development of various software systems in teams that are geographically dispersed in nature, where participants or developers collaborate across different time zones, cultural backgrounds, languages, and geographical distances (Bjorn, Soderberg & Krishna, 2017). Mohagheghi (2004) defines GSD as software work conducted across multiple geographical locations through synchronous and asynchronous interactions. Therefore, GSD can be understood as the "development of a software artifact across many different locations" (Vallon et al., 2018, pg. 162).

Coordination strategies, technologies and mechanisms are an important area of research within GSD, where the focus is on how developers are able to coordinate various tasks within software development projects, and how the overall organisation supports the coordination of activities (Boden, Nett, & Wulf, 2007). These activities should align with the goals of GSD projects that can be defined in terms of cost – deliver within budget and time – deliver projects on time, and lastly

quality – conforming to expected standards (Sriram & Mathew, 2012). Therefore, the main objective of GSD is to develop high quality products at a lower cost than co-located developments by optimizing the resources (Shrivastava & Date, 2010). Studies have shown that globally distributed agile teams do adopt various roles and activities, as needed, to facilitate self-organization during projects, however, it is rarely achieved and has not been reported in globally distributed development contexts (Licorish and Macdonell, 2021). However, many studies have reported on several GSD challenges that software organizations have experienced during the transition process to remote working. The most common challenges identified were mainly communication, coordination, control, and were a result of temporal, socio-cultural, and geographical distances (Alsahil, 2017).

2.1.4 The Shift to Remote and Distributed Work

Prior to the Covid-19 pandemic, remote work has been steadily on the rise but comprised a relatively modest share of the labour force. It was common for companies to have no remote workers or a small portion of remote workers (Brynjolfsson et al., 2020). According to the Ozimek (2020) report nearly half of businesses in the *pre-Covid future workforce survey* reported that none of their workers were working remotely with only 2.3% of managers who were hiring for remote workers, and only 13.2% of the represented labour force was fully remote in the Unites States.

The Covid-19 pandemic and the associated health and safety regulations that restricted mobility in almost all countries during 2020 and 2021 have led to the acceleration of remote work and forced many workers to resort to home working arrangements (Henry, le Roux & Parry, 2021). Although measurement and conceptual challenges hinder the results of key findings, there is evidence of an increase in work-related tasks being conducted remotely (Henry, le Roux & Parry, 2021). Allen, Golden, and Shockley (2015) found that two points substantiate the increase of remote work prior to the pandemic. The first is the percentage of employees who choose to work remotely, while the second is the percentage of organizations that promote or allow their workers to operate from home. The

American Community Survey (which measures the number of US employees who work remotely) found an increase in the number of these employees from 1.8 million in 2005 to 3.9 million in 2015[2]. Furthermore, a 2015 Gallup poll noted that 37% of American citizens have telecommuted for work and at least 24% do this at least half of their workdays (Jones, 2015).

To further support the impact of the Covid-19 pandemic on the rise of remote work, a recent survey from Slack HQ estimates that over 16 million knowledge workers in the United States were transitioned to remote work within the first few weeks of the declaration of the Covid-19 pandemic (Rudnicka et al., 2020). This number of remote workers will likely have a large impact on the future of work, with many organisations possibly implementing remote work for the long term. Similarly, another recent survey conducted by Brynjolfsson et al. (2020) found from a sample of the US population that half of the workforce is working remotely, including 35.2% who report they are commuting and recently made the transition to working from home.

The common thread when analyzing the above-mentioned surveys and polls is that there was an ongoing growth trend with an increasingly larger proportion of workers and organisations adopting remote working practices. However, due to the Covid-19 pandemic this trend has dramatically increased the adoption of remote work. This is evident in the increased number of individuals who are working remotely and has paved the way to the 'new normal' where a much larger population of workers will likely operate remotely (Henry, le Roux & Parry, 2021).

2.1.5 Information and Communications Technology and Remote Work

Information and communications technologies (ICTs) are frequently used to facilitate communication and collaboration in the absence of physical, co-located, face-to-face interaction. Furthermore, virtual teams use ICTs to mediate communication and to coordinate work systems (Henry, le Roux & Parry, 2021).

[2]https://www.flexjobs.com/blog/post/flexjobs-gwa-report-remote-growth/

Zhang, Aikman, and Sun (2008, pg. 628) define ICTs as "technologies used by people and organizations for their information processing and communication purposes".

According to Riemar, Steinfield, and Vogel (2009) the umbrella term "eCollaboration" refers to all ICT-based collaborations within and between organizations. The concept of eCollaboration specifically describes the practices of communication, coordination, and collaboration between people in distributed contexts such as virtual projects. Therefore, ICTs are the enablers of remote and distributed work. These digital technologies are a prerequisite for virtual teamwork.

It is important to note that, while ICTs offer more flexibility for living and work arrangements, they can also impose knowledge workers with distractions from their home environment and family members. In addition, these workers are faced with pressures regarding constant connectivity and responsiveness (Waizenegger, McKenna & Bendz, 2020). Ashforth et al. (2000) found that the nature of proximate work can be emotionally demanding and can ultimately lead to psychosocial implications of isolation and loneliness. At the individual level, Sewell and Taskin (2015) emphasize the need for remote workers to initiate frequent communication with their team members to create the impression of inclusion and comfort using ICTs. Therefore, the role of ICTs expands not only to serve as a mode of communication and collaboration, but also it can potentially reduce negative consequences such as job dissatisfaction, alienation of working from home by mediating organizational support (Waizenegger, McKenna, & Bendz, 2020). Furthermore, Henry, le Roux and Parry (2021) note that ICTs enable interpersonal connection - (a feature of distributed work) between co-workers within organizations that are mediated by technological infrastructures (e.g., enterprise social media).

Enterprise social media (ESM) have gained prominence as the key form of ICT in many modern organizations. Webber and Shi (2017) refer to a 2015 market analysis conducted by HootSuite that sampled 1600 global organizations and found that 72% are increasing their use of ESM. Despite their ubiquity, a large number of

definitions for ESM have been established. One of the most widely adopted definitions is the functional definition of ESM provided by Leonardi, Huysman, and Steinfield (2013, pg. 2). In this view, ESM are defined as:

> Digitized platforms that enable workers to (1) communicate messages with specific co-workers or send messages to everyone in the organization; (2) explicitly indicate or implicitly reveal particular co-workers as communication partners; (3) post, edit, and organize text and files linked to themselves or others; and (4) view the messages, connections, text, and files communicated, posted, edited, and organized by anyone else in the organization at any given time of their choosing.

Table 2.2 provides some examples of ESM tools and platforms used for collaboration (note, this table does not provide an exhaustive list, but merely an illustrative example). These ICTs provide a means of communication and collaboration amongst workers in companies, including instant messaging, file sharing (Anderson, 2016). These strong communication tools allow members within businesses and organizations to interact with external audiences such as product users, stakeholders (Wong, 2018). ESM technologies enable organisations or team members to exchange information, promote initiatives, and perform a range of tasks and activities (Anderson, 2016). This integrated social media is for *internal* communication and social interaction within the enterprise (Leonardi, Huysman, & Steinfeld, 2013).

Table 2.2: ESMs that teams can use to communicate and collaborate

Technologies/Platforms	**Description**
Slack	Slack is a team communication software tool that enables messaging in channels and direct messages either public or private and integrates popular services such as Google Drive (Dennerlein et al., 2016). Slack has gained wide popularity for ESM for organizations to increase the effectiveness of internal communication (Wong, 2018).
Microsoft Teams	Microsoft Teams supports communication (chat rooms) and collaboration (working simultaneously on files) and the service been used in many organizations around the

	world, particularly since the inception of the pandemic. MS teams is conceptualized as an ESM as it allows fast and informal communication (Lansmann, Schallenmuller, & Rigby, 2019). Important to note that MS teams are part of Microsoft 365 to allow organizations to access it through a subscription.
Jive	Jive is a provider of corporate social media technologies that facilitate business communications, connections, and collaborations amongst employees (Osch, Steinfield, & Balogh, 2015).

In addition to ESM many organisations or teams use other types of ICTs to facilitate communication and coordination. Among others, these include project management software like Trello and video conferencing software like Zoom. Trello is a project management solution that supports online collaboration and communication to accomplish project tasks and goals (Parsons et al., 2018). Technologies such as instant messaging or video conferencing are bound by time, however, recordings of the interactions do exist (Leonardi, Huysman, & Steinfield, 2013). Many video conferencing platforms are available such as GoToMeeting, Skype, Google Hangouts, Zoom.

2.1.6 Conclusion

This section has provided a high-level overview of remote and distributed work, with the work definition of remote work adopted as any work performed across multiple locations. In addition, this thesis will follow the work definitions of distributed work defined by Rockman and Prats (2015) that was described in Table 2.1 refers that remote or distributed work occurs when one is distributed from others and relies on communication technology to manage the physical distance. Through analyzing different surveys and data it is clear that there is an increase in remote work practices across various domains. Therefore, many companies rely on ICTs to mediate communication and collaboration particularly since the start of the pandemic. The discussion of how the increase of remote working has impacted agile software practices will be provided in the later part of this literature review.

2.2. Agile Development

This section aims to discuss the key aspects of agile software development and consists of the following content: first, an overview of software engineering will be provided, second, the historical background of agile development will be outlined, third, the core values of agile development will be discussed and, finally, the most commonly use agile methodologies will be described.

2.2.1 Software Engineering Methodologies

To understand the notion of software methodologies, Avison and Fitzgerald (2003) refer to information systems development methodologies as tools or procedures that are needed by software developers to complete assigned tasks. Similarly, in accordance with the British Computer Society definition, an IS methodology is viewed as "a recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management and training for developers of information systems." "A methodology should tell you 'what' steps to take and 'how' to perform those steps but most importantly the reasons 'why' those steps should be taken, in that order" (Jayaratna, 2004, pg. 37). There is also overlap between different definitions of these terms between software development and project management, however, for the purpose of this study the terms, methodology, methods, approaches, or procedures directly refer to agile development.

This thesis treats agile development as a software engineering (SE) and project management approach. The term "software engineering" can be understood to mean "an engineering discipline that is focused upon all aspects of software production from early stages through to maintaining the system" (Sommerville, 2010, pg. 7). In general, software engineers adopt a systematic and organized approach to their work. This systematic approach is known as the *software process* which is a sequence of activities that leads to the production of a software product (Rajib, 2014). Software products can be small to large-scale programs; however, the development of large software products is complex in nature and involves following

software engineering practices to achieve good quality software cost-effectively. Software engineering principles follow two techniques to reduce complexity, namely: abstraction - "implies that a problem can be simplified by omitting irrelevant details" and decomposition – "complex problems are divided into several smaller problems and then the smaller problems are solved one by one" (Rajib, 2014).

Agile development emerged as a reaction to previously developed SE methodologies – plan-based and traditional methods, respectively (Larman & Basili, 2003). Traditional project management methods such as the Waterfall Model follow a linear approach to software development, and in the recent past, evidence has shown that these traditional approaches cannot sufficiently cope in an ever-changing and unpredictable environment.

2.2.2 Establishment of the Agile Development Movement

This section will highlight the key events and influencers that played an important role in the agile development movement. To provide background and context for this study, the definitions and contents surrounding the establishment of the agile development movement will be briefly reviewed. The term "agility" was formally defined in agile companies and virtual organizations for flexible software development in 1995, as agility is dynamic, context-specific, growth-oriented, and aggressively change-embracing (Volkan, 2012). Alistair Cockburn argues that "agile implies being manoeuvrable and effective. An agile process is both light and sufficient. The lightness refers to staying manoeuvrable and sufficiency is a matter of being prevalent" (Abbas, Gravell & Wills, pg. 95, 2008). To understand the process and events that shaped the establishment of the agile development movement, this thesis will provide a high-level outline of some of the historical developments in software engineering. It is important to note that these developments were not presented chronologically. The historical evolution of software methodologies will be reviewed using the three-staged approach developed by Avison and Fitzgerald (2003) and expanded by adding an additional

stage. Avison and Fitzgerald (2003) refer to the three-staged approach: Pre-methodology era, early-methodology era, and methodology era.

Pre-Methodology Era

The pre-methodology era was common in many large North American and European organizations of the 1960s, characterized by a relatively ad hoc (unplanned activity which does not follow any documentation and test design) way of developing software and information systems. Therefore, no software development or project management methodologies were available, hence the concept of software development did not formally exist yet (Avison & Fitzgerald, 2003). At this time, the focus of most large organizations was driven toward solving technical problems through programming which was embedded in hardware. Many of these technical hardware problems stemmed from early programmers who often showed a lack of skills and training (Ehlers, 2011). Moreover, many IT departments had insufficient resources, thereby placing emphasis on maintaining existing systems, rather than the development of new software. This manifested into the popularity of these ad hoc or informal systems that matured as they were slowly integrated into business processes (Ehlers, 2011). The software was designed to never ensure reliable and scalable systems but to ensure they were functioning (Boehm, 2006).

Early-Methodology Era

The early-methodology era occurred in the 1970s and 1980s and is the foundation by which many methodologies are built around today. To enhance the quality of software systems and their development procedures, during the early-methodology era, numerous organizations initiated the construction of formalised models for software development. During the 1970s, IT organizations identified phases or steps of software development that were at the forefront of classical engineering and consisted of several “checkpoints.” This step-by-step approach is commonly known as the Systems Development Lifecycle (SDLC) or “Waterfall model” (Avison & Fitzgerald, 2003).

The SDLC is a process of building or maintaining software systems that typically includes various stages from preliminary development through to post-development software testing and evaluation (Leau et al., 2012). The development of the SDLC has important links with project management as emphasized by (Koskela & Howell, 2002). Both the SDLC and project management harmonize to form a complete methodology as, throughout the lifecycle, both methods work together to achieve business goals and user satisfaction[3].

Several different models describe various approaches to the SDLC, and these can be broadly classified into the following categories: linear, iterative, and a combination of both linear and iterative. A linear model is sequential where one stage, upon completion, inevitably sparks the initiation of the proceeding stage. In contrast, the iterative model ensures all stages to be re-evaluated post-development, hence, development remains a constant improvement throughout its lifecycle (Ruparelia, 2010).

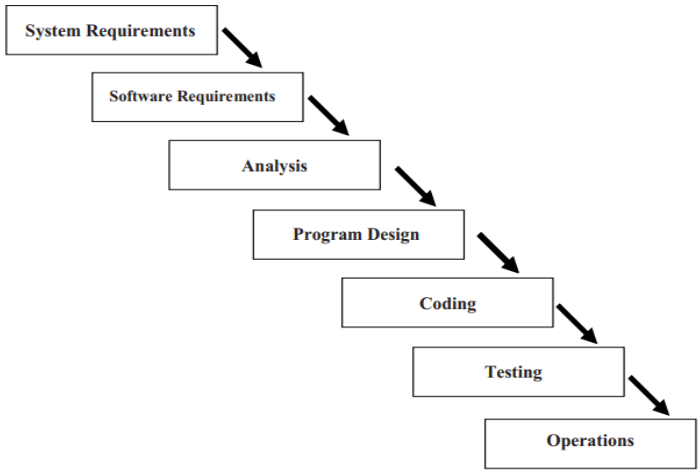


Figure 2.1: A classical view of Royce's 'Original' Waterfall Model (Casteren, 2017).

The SDLC or Waterfall Model depicted in Figure 2.1 was orchestrated by Royce (1970) in his well-renowned paper "Managing the Development of Large Software Systems" which called for the transition from informal or ad hoc methodologies

[3]The Project Management Method and the SDLC - The Ultimate Guide to the SDLC (ultimatesdlc.com)

towards formally established information system development methodologies. The Royce Waterfall Model was initially designed to teach software development and it was never intended to be a practically useful approach to sequential software development (Aitken & Ilango, 2013). However, given the nature of the approach it gained significant popularity amongst managers within many large organizations at the time and has since received widespread adoption across many different industries. One of numerous elements in the Waterfall Model is its requirements for documentation.

Methodology Era

The reality of the 1980s witnessed tremendous growth in the use of computer-based systems in organizations, and this had a strong correlation with the increasing amount of software development projects undertaken (Ehlers, 2011). Several different approaches to IS development emerged which gave rise to the methodology era. These methodologies can be classified into a few movements: 1. methodologies designed to enhance the traditional waterfall model and 2. the proposition of new methodologies that differentiate themselves from traditional linear models (Avison & Fitzgerald, 2006). During this period, many limitations of traditional approaches like the Waterall Model were experienced which ultimately led to criticism. Some potential limitations of these traditional approaches include: failure to meet management needs, instability, ambitious system design, inflexibility, user dissatisfaction and documentation problems (Avison & Fitzgerald, 2006).

As a response to the above-mentioned limitations of traditional approaches to software development, there was a need to search for alternative structuring of the SDLC. After thorough research and use of different methodologies, it became patently clear that the Waterfall Model and use of classical engineering would not be as viable for the software industry as anticipated because businesses and their environments change frequently (Ehlers, 2011). Many publications and authors emphasized the need to explore more flexible alternatives as they focused on changing and adapting the organization to meet consumer demands (Ehlers, 2011).

At the time, the up-and-coming methodology was *Iterative Incremental Development* (IID), which had been developing from the late 1930s work of Walter Shewhart and seeded NASA's early 1960s Project Mercury. This project highlighted several practices such as short iterations and test-first development which at the time were already adopted in the Mercury project (Larman & Basil, 2003). IDD projects encompass a preliminary specification stage and development teams performed short iterations with minimal feedback, ultimately producing releasable software (Clarke, O'Connor & Yilmaz, 2018). This reflexive or iterative process compensates to some extent for the changing demands in different environments during the development process. In addition to this, it nullified many criticisms against the Waterfall Model. Therefore, given the reality of globalization, many software developers soon realized the need for adaptive and people-orientated methods (Abbas, Gravell & Wills, 2008).

Post Methodology Era

The post methodology era led to the emergence of new software methodologies because several software development attempts throughout the 1990s and 2000s were dissatisfied with existing software methodologies. During this era, methodologies were the panacea to many problems and limitations of traditional development approaches, and they were often chosen and adopted for the wrong reasons. Some organisations simply wanted a better project control mechanism, others more user involvement (Avison & Fitzgerald, 2006).

By the early 2000s, Kent Beck (an American software engineer and one of many original signatories of the Agile Manifesto) convened a meeting in Oregon that consisted of various key figures, namely: Alistair Cockburn, Martin Fowler, Ron Jeffries, and Robert Martin. During this meeting, these practitioners collaborated and expressed their support for a variety of "Light" methodologies, but nothing manifested. During this period, an array of articles was published that referred to the category of "Light" or "Lightweight" methodologies (Highsmith, 2001). These methodologies refer to "Agile Methodologies" that embrace practices that allow

developers to build solutions efficiently and quickly, with better response to change and focuses on short development lifecycles (Khan, Qurashi & Khan, 2012).

In February 2001 in Utah, a group of different practitioners with a strong understanding of the software industry cooperated to establish what is called the "Summit" and, following this meeting, published the Manifesto for Agile Software Development. At this time, the participants decided to use the term agile to refer to these newer methodologies, namely: less documentation driven and lightweight software development processes." (Highsmith, 2001).

The importance of this meeting was to reach common ground and agreement on the Manifesto for Agile Software Development. The primary goal of the agile manifesto was to set the philosophy of agile software development, and to create awareness of the idea to produce high-quality, valuable working software, and ensure development teams adhere to Kent Beck's four core tenets of value (Corbucci et al., 2011). These four core tenets of agile development are listed below:

1. **Individuals and interactions** over processes and tools

2. **Working software** over comprehensive documentation

3. **Customer collaboration** over contract negotiation

4. **Responding to change** over following a plan

In each tenet, the item on the left is favourable over the classical or traditional engineering discipline on the right. At the summit, the attendees expanded on Kent Beck's four tenets of value and described the twelve principles of agile development, which are provided below:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Businessmen and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

2.2.3 Core Values of Agile Development

To further understand the core characteristics of the central values or tenets, these will be elaborated upon in this section in more detail. The four tenets of the Agile Manifesto are at some point interrelated, and one should consider them coherently with design approaches for a particular project. In each tenet, the item on the right is heavily process-based with emphasis on heavy documentation. The limitations and drawbacks of these methods were realized by many software developers (Ehlers, 2011). Some of the perceived limitations of these methods included: difficulty to grasp, labour intensive, time-consuming, and thus significantly affecting the software development process (Misra et al., 2012).

2.2.3.1 **Individuals and interactions** over processes and tools

The first tenet of the agile manifesto indicates that people are more important compared to certain processes and tools. This first key shift in emphasis was proposed as a response to "command or autocratic" project management practices that were viewed as relatively impersonal to people, and rigidly defined processes where the "process is largely responsible." (Cobb, 2015, pg. 22). From a project management perspective, this tenet calls for key decisions to be made by developers with an emphasis on empowering people to perform their jobs which leads to a greater community between project members, ultimately manifesting in close team relations, boosting team spirit, and flattening hierarchies (Cobb, 2015).

A common thread running through agile development points out the fact that individual interactions (the channel of communication) are core to the discussions between people to discover new solutions. Therefore, emphasis on human involvement in software development matters. Highsmith and Cockburn (2001) support this idea by indicating their preference for undocumented processes with meaningful interactions that are crucial for project success over documented processes with minimal interactions. Autonomy, decision-making, and individual responsibility afforded to developers in an agile approach is substantially more than what would traditionally be available in earlier approaches to software development.

Ehlers (2011) refers to the quality (this is the knowledge, skills, and abilities) of individuals involved in the agile development project to be largely responsible for the election of development approaches. Therefore, the transition from prepared planning to an individual, assumes the capability of an individual is well within reach of coping with added responsibilities. This would require some highly skilled or (above-average) developers to manage agile projects compared to the skills requirements typically needed in more linear projects (Ehlers, 2011). However, Duncan (2019) affirms that if there is effective communication, collaboration, and trust among people, they are most likely to overcome many limitations within the process, tools, and environment. The main challenge to ensure this in a work

environment is that many teams are distributed across both distance and time boundaries. However, people who recognize the distribution across large distances and time use technology to facilitate computer-based communication using Skype, Google Hangout, and more (Duncan, 2019).

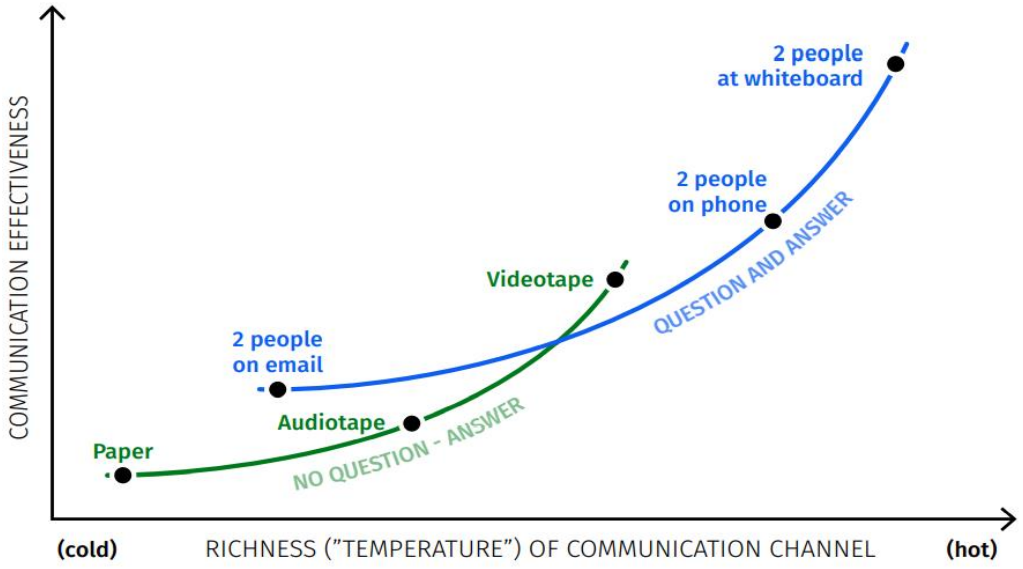


Figure 2.2: Measure of Quality in *Computer-mediated* Communication (Duncan, 2019). Original work (from Cockburn, 2002; 2006)

Computer-based communication can be quantified to determine its level of effectiveness. Cockburn (2006) argues that the richer the communication media, the more effective it is. The above illustration is from Cockburn's book "Agile Software Development: the cooperative game" and captures the discoveries of researchers, such as McCarthy and Monk (1994) in measuring the quality of computer-based communication. In relation to Figure 2.2, the different media are classified into two broad categories, "Question and Answer" and "No Question-Answer", based on the media's capability to facilitate two-way, interactive communication.

The question-and-answer (blue line) category encompass all real-time and two-way media except for e-mails, which are clearly capable of two-way communication but

lacking real-time engagement. In contrast, the no question-answer category (green line) consists of all media that do not allow for two-way communication, thereby the receiver is unable to provide direct feedback to the sender (Svalestuen et al., 2017). The vertical axis indicates the effectiveness of communicating information while the horizontal axis shows the richness of the communication channel. Cockburn (2002) argues that "hot" communication provides more information than "cold" media. Coherently, these axes represent the time and energy to communicate information between people. An important observation to note is where text-based documentation sits ("Paper" on the diagram) compared to face-to-face communication ("2 people at whiteboard").

The last part of the "individuals and interactions over processes and tools" tenet is defined by "processes and tools" which potentially implies that processes and tools are less important in agile projects. Several reasons support that this is not the case: agile uses well-defined processes like Scrum (see Section 2.3.1) and is designed to be integrated into business processes rather than adhere to some rigid process. Therefore, tools play an essential supporting role by leveraging and facilitating human interactions within the business environment (Cobb, 2015). Moreover, in accordance with Clarke, O'Connor and Yilmaz (2018) it can be argued that the manifesto itself shows signs of misfit with emerging agile practices, especially in its notion of "individuals and interactions over processes and tools" because the use of processes and tools is now a central concern. Therefore, while individuals and interactions are important, much of remote and distributed work is driven using certain processes and tools because face-to-face communication is not always possible.

2.2.3.2 **Working software** over comprehensive documentation

The second tenet of agile development is "working software over comprehensive documentation." This tenet was proposed in response to typical traditional approaches which placed emphasis on the production of documentation (Ehlers, 2011). Working software can be defined as the agile approach to quality which expects each iteration of work to be of software product quality, ensuring defect-

free software. To be fully useful in production, all associated user manuals, installation procedures should be completed. From an agile perspective, the "software" is the user documentation and is considered part of what must "work", as well as the code itself (Duncan, 2019). The "golden" principle is that working software has more value than extensive documentation, it allows the project team to monitor the production rate of software and provides quick feedback to bolster user satisfaction. Highsmith and Cockburn (2001) describe "the unforgiving honesty of working code" as one of many fundamental concepts of agile development: Working code informs project team members about software requirements – as opposed to what will the requirements be.

One of the main problems associated with heavy documentation is that it can inhibit face-to-face communication. The traditional Waterfall Model focused on heavy documentation. Therefore, project members would develop requirements specifications and software testing was performed against meeting that specification without necessarily directly communicating with each other. Additionally, in many traditional approaches, end-users had limited involvement. This led to several opportunities for problems: the difficulty of defining detailed requirements in a project and relying on documentation can result in the lack of communication about certain requirements that ultimately delivers software that does not meet user satisfaction (Cobb, 2015).

The importance of effective communication during software development is paramount because this allows programmers to communicate with software by using it in different ways. Software responds when given the specification, therefore, software developers can monitor and validate the software through forms of testing and inspection. If this is done frequently, this communication can decrease the risk of producing software defects, ultimately satisfying customer expectations (Duncan, 2019). It is important to note that this tenet does not imply the eradication of formal documentation in an agile project. Rather, the key factor is that any documentation should provide value to the software development

process and should never be fully omitted and the emphasis should be placed on working software (Cobb, 2015).

2.2.3.3 **Customer collaboration** over contract negotiation

Collaboration with customers is focused on the "what" side of work rather than the "How." That is, it highlights the customer needs as different software specifications. Highsmith and Cockburn (2001) define customer collaboration as the active involvement of all stakeholders on a given project, namely: the sponsor, customer, and developer who share common responsibility and expertise.

Agile development prioritizes customer collaboration over contract negotiation because contracts typically seem to be based on the belief that the requirements need to be defined upfront, which does not promote a collaborative relationship between people (Duncan, 2019). However, Abrahamsson and Ronkainen, (2002) refer to the negotiation process itself which is a means of creating and maintaining a viable relationship between both developers and customers. Therefore, this implies that contract negotiation has some relevance and is not entirely excluded from agile approaches.

Typically, project managers have been responsible for controlling costs and deliverable deadlines, doing this requires some form of contract to deliver software-based on a defined specification. Of course, this requires some level of change control to monitor the changes in the requirements as the project develops. Therefore, the importance of the contract increases significantly with the size of the project (Abrahamsson et al., 2002).

Agile methodologies recognise that in an unpredictable environment, a more collaborative approach can be more efficient – as opposed to having contracts with predefined or upfront requirements. These fixed upfront requirements limit software developers' abilities to completely change project methodologies, therefore, detailed requirements could not be further elaborated as projects develop (Cobb, 2015). To collaborate with customers positively, more rigorous contracts will not be beneficial. It is more viable to create a general agreement or

collaboration of high-level requirements between the project team and end-customer, ultimately this will be delivered within a time and budget. (Cobb, 2015).

By involving the customer directly in the development process, many positive occurrences and adaptations can be realised. The constant communication between the project team and customer assists developers to produce products that satisfy customer expectations (Ehlers, 2011). Contracts with predefined requirements would certainly endure limited communication between project members and customers, hence, ongoing collaboration between customers are not prioritized. Therefore, it is imperative that the customer sees the value in frequent, open communication in terms of their needs and what development can be ascribed to as "working" (Duncan, 2019).

It is important to recognize that this tenet is relative and will be applied differently in different situations. At one extreme, an agile approach within a government-contracting environment (such as capability maturity model integration – CMMI which is used around the world to build scalable, high-performance organizations to deliver the promises of agile approaches) requires contracts because of deliverables, costs, and milestones, however, customer collaboration is still valued within this environment to some extent. At the other extreme, projects exist where the requirements are uncertain, hence a much more collaborative approach is required with the customer to define the requirements as the project unfolds, without the need of a contract (Cobb, 2015). Therefore, customers are an important source of knowledge and the initial idea of customer management is based on the notion – *I work for the customer*.[4]

2.2.3.4 **Responding to change over** following a plan

Nothing represents the traditional notion of "agile" more than this tenet. A typical definition – "able to move quickly and easily" which implies the ability to change with relatively minimal effort (Duncan, 2019). This final tenet in agile software

[4]https://www.flashover.blog/posts/customer-collaboration-over-contract-negotiation

development indicates a preference for "responding to change over following a plan," where its focus is adaptability to change during software development. Ehlers (2011) refers to project teams and customer representatives who constantly adapt to stay current and relevant with realities changing requirements as opposed to changing management processes that is followed by traditional methodologies. Therefore, this tenet is in response to many projects that are centered around controlling schedules and costs. This limits customer involvement to change the requirements in terms of the scope and cost of the project (Cobb, 2015).

The issue in adopting this tenet in an environment where the requirements are uncertain is that it pressurizes users to define the requirements upfront without envisioning the result. It is more effective to recognize and accept at any given moment that the requirements will change as the project envelops and respond by developing an approach around that change (Cobb, 2015). Responding to change requires a meaningful yet responsible approach that directs that change towards greater stability in the results. From a software perspective, it implies producing a satisfactory, useful delivery to end customers and no fixed upfront requirements (Duncan, 2019).

The implementation of responding to change requires constructing a process for working that allows for change with reduced cost and risk, and at the core of this approach, is the iterative development paradigm (Faware, 2002). By recognizing the change and key presuppositions of the development process, it becomes easier to enforce adaptability as well as shortens the life cycle of the project (Ehlers, 2011). Many research emphasize the difficulty and inefficacy of long-term, detailed planning when working in an environment where change is inevitable and result predictions are limited. However, Duncan (2019) supports the act of planning within the agile team, as it enables communication and collaboration which can build trust and confidence in the workspace to gain a consensus of the headed direction.

Boehm's life cycle cost differentials theory in software engineering economics, explains that throughout the project's development the cost of change increases,

therefore, to minimize the cost it is essential to make changes when the problems become known (Highsmith & Cockburn, 2001). An agile team that conforms to a plan depends on the individual and their creativity to solve problems as they become apparent. Creativity holds higher value compared to written rules and is the only way to manage sophisticated software development issues in diverse situations (Highsmith & Cockburn, 2001).

2.3 Agile Methodologies

To extend the high-level, abstract principles and tenets of agile development, and provide a concrete description of how these principles have come to be implemented, this section will describe the practices applied in some of the most commonly adopted agile development methodologies. Additionally, while the primary focus of the section will be placed on the most popular methodologies, less frequently used agile methodologies will also be briefly reviewed in this section. In the course of these descriptions, to provide context for the development of these methodologies, the events that cultivated and precipitated the emergence of these approaches will be briefly explained.

Agile software development methodologies include a variety of popular methodologies, including Extreme Programming, Scrum, and Kanban (Dingsoyr et al., 2012). The practices inherent in these methodologies encourage user and stakeholder interaction, thus, guiding the production of the end software product or service. Additionally, all of these different methodologies aim to support the delivery of software to users through short iterations or pieces of work (Dingsoyr et al., 2012). Since the inception of agile during the early 2000s, the adoption of agile methodologies in organizations has grown and become a viable alternative to traditional development practices (Ehlers, 2011).

Preceding 2009, many IT projects solely utilized agile methodologies for their holistic and applicable attributes (Stare, 2014). For example, numerous international firms such as IBM, Nokia, Microsoft, Google, make use of agile practices to produce software (Gandomani et al., 2013). Therefore, business

environments recognize agile methods to be based on leadership and exhibit frequent user collaboration. The upside is innovation and collaborative decision making (Gandomani et al., 2013). Due to the complexity of changing business environments, an organization's agility is no longer a requirement, but a condition to access or remain on the market. An agile enterprise adapts fast to user demands and recognises market opportunities (Stoica, Mircea, & Ghilic-Micu, 2013).

From this, one can gather that agile methodologies are ideal for projects that require high variability in deliverables due to changing user requirements, capabilities of people, and new and evolving technology being used (Nerur, Mahapatra & Mangalaraj, 2005). In the following sections, three popular agile methodologies will be outlined. This will commence with Scrum followed by eXtreme programming (XP) and Kanban. It is important to note that while these are distinct methodologies, in many cases 1) organizations/teams adapt them to their own situations and 2) combine them with aspects of other methodologies.

2.3.1 Scrum

The agile methodology termed "SCRUM" is a management framework by which people can manage complex adaptive problems (one that changes when attempting to solve it before the solution is completed), used since the early 1990s. Scrum is not a process, rather a framework grounded in the principles of agile development that aims to ensure the delivery of products of the highest possible value using various techniques and processes (Schwaber & Sunderland, 2017).

Scrum is the agile development methodology with the most reported cases of being used within companies today. From a 2019 scrum master trends report published by scrum.org, one of the key findings from the pool of 2100 participants across 13 countries was that 81% were using scrum coherently with other agile practices like Kanban[5]. These statistics on scrum adoption are support by several websites that refer to the findings of the scrum master trends report of 2019[6].

[5]2019 Scrum Master Trends Report Published (infoq.com)
[6]Scrum Master Trends Report 2019 - DZone Agile

Sunderland and Schwaber (2007) acknowledge the original scrum research paper by Takeuchi and Nonaka (1986) as the bedrock for coining this development approach 'Scrum.' The original research paper, *The New, New Product Development Game*, came about to address the reaction to the fast-paced, competitive commercial world of product development in the 20th century. The new emphasis was on speed and flexibility which called for a new approach for new product development. Takeuchi and Nonaka described the adaptive, self-organising and rapid product development in Japan and compared it with the game of rugby. This analogy refers to when rugby players must devise a strategy to bring an out-of-play ball back into the game using teamwork.

Building on Takeuchi and Nonaka, the approach was developed by Jeff Sunderland at Eastel Corporation in 1993 (Sutherland, 2004). This approach brought the solution to the challenge of keeping abreast with ever-changing technology development due to the iterative nature of Scrum (Paul & Behjat, 2019). Scrum in essence involves a small team of people. The individual team is highly flexible and adaptive. These strengths continue operating in single, several, and networks of teams that create, release, operate and sustain the work and work products of many people involved.

Scrum is an iterative process of developing products and software where all team members should function to produce the system flexibility in a changing business environment (Awad, 2005). The Scrum work phase is based on iteration, one work unit is called a Sprint. The goal of the sprint is to develop releasable units of software (Popli & Chauhan, 2011). Sprints usually undergo 1–4-week cycles and new software functionality is demonstrated at the end of each sprint. Despite the general emphasis in agile development on avoiding processes and tools, scrum is implemented through scrum roles, artifacts and events that are maintained throughout the scrum methodology. All elements of the Scrum Framework be further expanded in the upcoming sections.

2.3.1.1 Scrum Roles

The scrum team consists mainly of the product owner, development team, and scrum master, respectively. The composition and interrelatedness of these roles are crucial and contribute to the effectiveness of scrum patterns (Hundermark, 2015). These roles will be further elaborated on within this section.

Product Owner

The central point of product leadership resides with the Product Owner. This individual is responsible for maximizing return on investment by recognizing product features and affirming Scrum team members to prioritize the most valuable product features for the next sprint (Sunderland, 2010). The primary role of the product owner is to manage the product backlog. In addition to this, the product owner is responsible for maintaining and communicating with team members to achieve the right product for the customers (Rubin, 2012). It is important to note that product owners have several responsibilities, and for the product owner to succeed, the interaction and commitment of the entire organisation or team is paramount. The product owner merely represents the desires of a committee in the product backlog (Schwaber & Sunderland, 2017). However, the product owner does not instruct teams that are self-organizing and self-managing.

Development Team

The development team is comprised of various professionals who perform the delivery of releasable increment “done” products at the end of each sprint. (Schwaber & Sunderland, 2017). Therefore, the primary goal of the development team is to focus on efficiency by satisfying product owners and users through the delivery of expected products (Hundermark, 2015). The ideal scrum team size is between 5 to 9 people to make for more interactive communication; its members are responsible for working collectively to produce quality working software (Rubin, 2012). Scrum teams are self-organizing and cross-functional. Self-organizing teams choose the best way to accomplish their work without any

direction outside the team, while cross-functional teams have all competencies to accomplish their tasks such as working software (Schwaber & Sunderland, 2017).

ScrumMaster

The ScrumMaster in agile is known as the servant leader, as he/she has the responsibility to manage the team, however, not to the full responsibility like a Project Manager (Cobb, 2015). Therefore, the ScrumMaster is not the manager of the team but serves to educate and guide the product owner, and the team in the use of Scrum by ensuring the team understands and adheres to the practices of Scrum (Sunderland, 2010). Furthermore, a ScrumMaster protects the Scrum team from interruptions and demands from stakeholders, this ensures smooth operation of Scrum practices. These interactions are from the ScrumMaster which ultimately fosters greater value within the Scrum team due to frequent monitoring and feedback (Schwaber & Sunderland, 2017).

2.3.1.2 Scrum Artifacts

As part of the Scrum Framework several tools and documents (called artifacts) are used.

Product backlog

In many traditional approaches the functionality of a product must be determined in detail before the design and development process begins. In contrast, agile approaches are different because the functionality of the product changes during the development process.

In Scrum, the product backlog contains an ordered list of features and requirements needed in the product. These items are prioritized based on their value towards customers (Ehlers, 2011). Many new items will be added overtime, existing items are converted into smaller items and some items are removed as the desired feature becomes no longer needed. Therefore, the product backlog is a living document that needs constant refinement to keep it useful and current (Hundermark, 2015). At any point, the product backlog is the single, definitive view of “everything that could

be done by the team ever, in order of priority". Only a single product backlog exists; this means the product owner is required to make prioritization decisions across the entire spectrum (Sunderland, 2010). The level of detail required by the product backlog is managed and determined by the product owner. Additionally, the product owner ensures that the product backlog items are placed in the appropriate sequence from high-value items, going at the top of the product backlog and low-value items at the bottom (Rubin, 2012). Therefore, the entire product backlog is the responsibility of the product owner.

Sprint backlog

Development teams visualize the sprint backlog by creating a *task board.* This is a physical representation of the work needed to achieve during the current sprint. The task board is an example of a Kanban board (see Section, 2.3.3) and informs the Scrum team about the planned work and its relevant status (Hundermark, 2015). The sprint backlog is a plan that changes with progression and as the development team works through the plan, they learn what is needed to achieve the sprint goal. As new work is required, the development team adds it to the sprint backlog and only the development team can change its sprint backlog during a sprint (Schwaber & Sunderland, 2017).

Scrum Burndown Chart

The Scrum Burndown chart is a visual tool used in Scrum. It is a visual representation that shows an estimate of how much work (measured in person-hours on the Y-axis and remaining time shown on the X-axis) is needed until the tasks are completed. This downward sloping graph aims to reach "zero-effort" for each sprint, hence, the name burndown chart (Sunderland, 2010). The burndown chart is not mandated in the Scrum process; however, the development team is responsible for managing their progress to achieve the sprint goal. Therefore, the burndown chart is designed in way to monitor the teams progress, and whether or not the development teams satisfies the sprint goal (Hundermark, 2015). Members of the

team can monitor its progress against a release plan by updating a release burn-down chart at the end of each sprint (Popli & Chauhan, 2011).

Increment

The increment is the total sum of all the product backlog items completed during a sprint as well as the customer value of the increments of all previously done sprints. At the end of each iteration, the new increment must be "completed", hence the increment must be useable and satisfactory enough for the Scrum team (Schwaber & Sunderland, 2017). The development team will present this at the Sprint Review (see Section 2.3.1.3) and the product owner will determine when to release it (Hundermark, 2015).

2.3.1.3 Scrum Events

Scrum is implemented through a series of events used during the Scrum process to create regularity and minimize the need for Scrum meetings. These events are time-boxed events, meaning every event has a maximum duration. Furthermore, these events are designed to enable critical inspection and transparency during the Scrum process. Any Scrum process will have at least a Sprint Planning Meeting, Daily Scrum Meetings, a Sprint Review Session, and a Sprint Retrospective Session.

Sprint Planning Meeting

The initial step toward imitating a new sprint is the sprint planning meeting. This plan is developed by the collaborative work of the Scrum team (Schwaber & Sunderland, 2017). Sprint planning answers two questions, each with a distinct purpose, namely: "What can we deliver by the end of this Sprint?" and "How will the work be conducted?" The development teams ask the first question to grasp the requirements in sufficient detail to forecast what can be delivered during the sprint (Hundermark, 2015). The second question addresses the development of a high-level design of features in which the team commits to complete by the end of the sprint (Sunderland, 2010).

Daily Scrum Meeting

During a sprint, a meeting is held daily. The development team members hold a Scrum time-boxed that are 15 minutes or less which is an inspect and adapt activity, also commonly known as Daily Stand-up because of the common practice of the team standing during the meet to help promote succinctness (Rubin, 2012). This meeting is basically a check-in for the team to coordinate and monitor the Scrum team's progression and to identify any challenges that may hinder progress (Cobb, 2015).

Cobb (2015) describes how each team member typically answers three questions:

1. What did you accomplish yesterday?

2. What are you going to accomplish today?

3. What obstacles are in your way

Using the answers to the above questions, the ScrumMaster can try to eradicate any obstacles to the process and is able to address any lingering member conflicts (Ehlers, 2011).

Sprint Review Session

A sprint review is held at the end of each sprint with the purpose of inspecting the increment and adaptation of the product backlog. Therefore, the key idea of the sprint review is based on inspecting and adapt activities. During the sprint review, the development team and stakeholders work together to evaluate what was done in the sprint (Schwaber & Sunderland, 2017). Therefore, the sprint review session is the key point in the feedback cycle that ensures product development (Hundermark, 2015). The most important element of the sprint review is an in-depth conversation and collaboration between the team and product owner to learn the situation and the market (Sunderland, 2010). A successful sprint review encompasses the inclusivity of not only the product owner in the decision-making process but also the collaboration between business users and other stakeholders to promote bidirectional information flow of review results (Rubin, 2012).

Sprint Retrospective Session

Upon completion of the review meeting, the ScrumMaster sets up a retrospective session in which team members review and discuss what was successful and unsuccessful during the sprint (Ehlers, 2011). This gives the development team an opportunity to reflect and identify opportunities for process improvement in the next sprints or proceeding sprints (Cobb, 2015). This session is usually a three-hour meeting for one-month sprints, however, for shorter sprints, the duration is decreased (Schwaber & Sunderland, 2017). After the retrospective session is completed, the entire cycle is repeated – starting with the next sprint planning session. Topics for discussion include questions such as "what is currently working?" and "what can be improved?"

2.3.2 Extreme Programming

In recent times, Extreme Programming (XP) is used in many use cases and applications such as web development, game development and has been successfully adopted across many global organizations. XP pre-dates the agile manifesto and was originally formulated by Kent Beck in 1996 who refined the development methodology in 1999 when he wrote the book - "Extreme Programming Explained" (Harrison & Labs, 2003). Kent Beck proposed XP as a reaction to the problematic nature of traditional methods, being long and heavy development cycles that yielded limited flexibility (Volkan, 2012). XP is significantly different from Scrum in that it has no given set of procedures, and therefore, aligns more closely with the original values of agile development (Ehlers, 2011).

XP is a framework of the agile software development model and its primary aim is to produce high-quality software. This ultimately stems from the large focus on demanding requirements from customers. Therefore, this stresses the importance of continuous communication between the developers and stakeholders (Yasvi, 2019).

The core of the XP process can be characterized by short development cycles, incremental planning, feedback, communication, and evolutionary design (Awad,

2005). This process allows the development team to define their own processes according to their unique situation (Dalalah, 2014). According to Williams (2003), XP team members allocate time for programming, project management, design, feedback, and team-building many times each day. The term "eXtreme" is derived from taking these principles and practices to extreme levels (Awad, 2005). Extreme programming follows 13 terms and practices, these are summarized in Table 2.3.

Table 2.3: *XP Terms and Practices*

Planning Game	In XP, the scope of the next release is planned according to certain priorities and specifications. Furthermore, XP planning is continuous and iteratively orientated by considering the schedule of short releases and goals for the preceding releases, according to customer requirements (Volkan, 2012). Therefore, the customer drives the development team's outcome.
Small Releases	The development team is responsible for the small frequent releases of working software for its customers to evaluate. The first release includes a small set of features while subsequent releases include newly added features (Dalalah, 2014). Small releases are important for both the development team and customer as it provides a form of feedback.
Metaphor	A document that describes the working of the system and expresses the project's vision that is aligned with the system's scope and purpose (Yasvi, 2019).
Simple Design	XP development teams build software using a simple design, they follow simplicity, and the design is exactly suited for the current functionality of the system (Lindstrom & Jeffries, 2004).

Tests	XP development teams use an automated unit test framework that writes tests for a new piece of functionality, before the functionality is implemented (Sommerville, 2010). Ehlers, (2011) refers to test-first programming, whereby test cases are developed upfront and should be automated and integrated into the software development process.
Refactoring	Refactoring is the process of improving the structure of code without compromising its function, hence, keeping the design of code simple and understandable to avoid duplication (Volkan, 2012).
Pair Programming	Pair programming is one of the practices in XP, whereby each pair of programmers work collaboratively to develop certain system functionalities, this ultimately increases software quality (Dalaha, 2014).
Continuous Integration	The XP development teams keep the system fully integrated to minimize the problems on a software project which is crucial for producing working code (Lindstrom & Jeffries, 2004).
Collective Ownership	The pair of programmers work on all the areas of the system, to create joint responsibility of the code (Sommerville, 2010).
On-site customer	The one-site customer acts as a bridge or agent between other customers and programmers, therefore, the customer leads the project to success (Xu, 2021).
40-hour weeks	This is the practice in XP which ensures that the development team in a project should work the number of hours that are sustainable to constantly deliver quality software (Yasvi, 2019).
Open Workspace	This is the physical space in which the development team operates within, generally includes small cubicles and computers for pair programmers (Beck, 1999).

Just Rules	As part of the XP development team, each member is mandated to sign up to follow the rules, however, these are subject to change if there is a general agreement between team members (Beck, 1999).

2.3.2.1 The Four Values of XP

The guiding values of XP are identified as being conducive to successful project development and to alleviate any complexities or challenges from the software development process, these values include: communication, simplicity, feedback, and courage.

- *Communication*: Sound communication is one of the key factors for the success of software projects because it enables customers to communicate their requirements to the developers, and between developers about ideas and design (Dudziak, 1999). Thus, communication stimulates efficient team functioning.
- *Simplicity*: The development team should aim for effective and elegant solutions for programming challenges to ensure the code is error-free and avoids duplication (Ehlers, 2011).
- *Feedback*: XP is a feedback-driven process; therefore, feedback is expected at all levels from customers to developers. Feedback has two important traits, 1. Quality and 2. Time, these are extremely relevant to the software development process as it provides a means of monitoring progress (Dudziak, 1999).
- *Courage*: The value refers to the morale of the development team and their ability to change or discard code to work with constant changes to their specification. Thus, the team must be willing to commit to find the correct solution, despite more effort being involved (Beck, 2004).

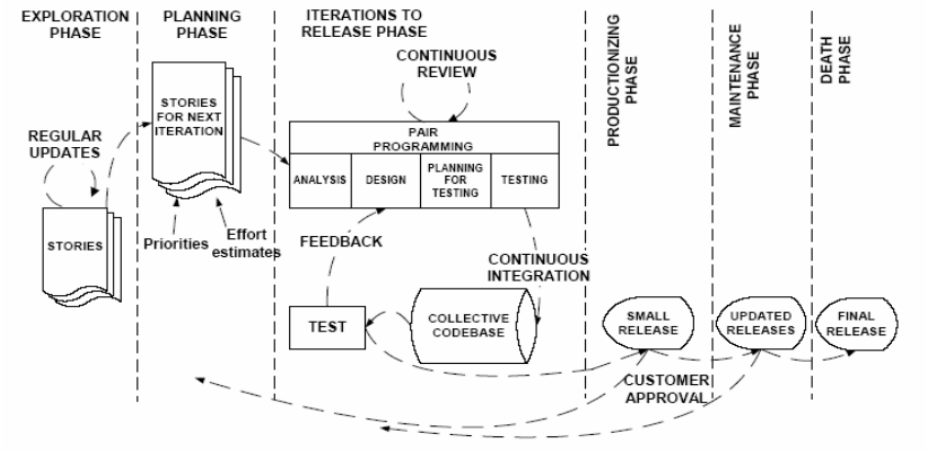


Figure 2.3: XP Lifecycle – Expanded View (Abrahamsson et al., 2002, p. 19)

2.3.2.2 Processes in Extreme Programming Lifecycle

The lifecycle of an ideal XP project, shown in Figure 2.3, is divided into six distinct phases and was introduced by Beck (2000). The following will describe the six phases of the XP lifecycle model:

Exploration phase

The XP lifecycle begins with the customers writing *story cards* that outline the requirements for the release of the product. This phase can take between a few weeks or months with the development team discussing potential technologies and architectures for the project (Ehlers, 2011). Furthermore, at some point, the team familiarizes themselves with the technology and tools used in the project to develop prototypes around possible architectures (Abrahamsson et al., 2002).

Planning phase

The exploration phase leads into the planning phase where a list of priorities is set for each user story and a schedule for the first release is developed (Awad, 2005). Therefore, the main purpose of planning is to ensure both customers and developers agree to a date by which the most valuable user stories will be done (Beck, 2004). The plan for the first release should be between two to six months, respectively.

Iterations to release phase

This phase refers to a set of functional test cases that must run at the end of each iteration (Juric, 2000). Furthermore, the result of product development will entail several iterations before it is production ready. Each iteration should last between one to four weeks to be successfully completed (Ehlers, 2011).

Productionizing phase

During this phase, additional checking and testing are done to monitor the performance of the system before the system releases the product to its customers. Moreover, at this time new changes may crop up; therefore, the team must carefully decide whether to implement these new changes into the current release (Abrahamsson et al., 2002).

Maintenance phase

Beck (2004) describes the maintenance phase as normality during the XP project because during the lifecycle the production of new functionalities, monitoring of existing systems, refractor of new technology, recruiting new project members, and farewell to members who move on, all happen simultaneously.

Death phase

The final phase is when the customers have no more user stories to implement, and all required documentation of the system is formally written as no further alterations to the design, code, and architecture are made, once the project enters the Death Phase (Awad, 2005). Moreover, the project can also enter the Death Phase if further development becomes too costly or if the product is seen as obsolete (Ehlers, 2011).

2.3.3 Kanban

Kanban is significantly different from the two proceeding methodologies discussed because it uses certain visual tools and techniques in software development. The methodology which is based on the Japanese word *Kanban*, which translates to “visual card or record” has become relatively prominent (Akturk & Erhun, 1999).

The Kanban system has been widely implemented as a *Just-in-time* (JIT) philosophy and traces back to the late 1940s and early 1950s (Gross & Mcinnis, 2003).

In a Kanban system, cards (either physical or increasingly digital) are used to produce and transport a given amount of information. This information processing is based on the shop-floor control system of the JIT philosophy. The shop-floor materials are controlled using cards, schedule sheets, production orders, product structures, or material lists (Junior & Filho, 2010).

The concept of the 'pull' system is also referred to as the Kanban system. In a pull system, the production of the current stage is dependent upon the demand of subsequent stages, for example, the preceding stage must produce the exact quantity based on the subsequent manufacturing stage (Huang & Kusiak, 1996). In contrast, push production systems have no explicit limit on the amount of work in progress. A Kanban system has a fixed limit on work and cannot exceed the amount of information that is allowed by the number of Kanban cards, hence, this limit is explicitly defined[7].

Kanban was created to fulfill the specific needs of the Toyota Production System to efficiently operate under market and production conditions. Therefore, Kanban is built for smooth and continuous delivery of customer value by carefully controlling the flow of quality work to resolve and discover issues (Brechner, 2015). A well-defined Kanban system has visual indicators that enable supervisors and managers to see the scheduling process – operators produce products on the actual usage as opposed to forecasted usage (Gross & Mcinnis, 2003). It is important to note that many works use the term Kanban indiscriminately meaning both "card" and "the system."

[7]Roser (2015). The (true) Difference between Push and Pull available at https://www.allaboutlean.com/push-pull/

2.3.3.1 Kanban Values

The Kanban system is value-based. It is motivated by the belief that the success of an organization is centered around the respect of all individuals who contribute collectively to the organization. These values embody the core of Kanban to improve service quality delivered by collaborating teams, therefore, Kanban cannot be implemented authentically without the adherence to these values (Anderson & Carmichael, 2016). These values of Kanban are summarized in Table 2.4[8].

Table 2.4: *The Values of Kanban*

Transparency	This is the belief in sharing knowledge openly to improve communication within an organization. The key focus here is clear-cut communication between individuals to avoid ambiguity or miscommunication.
Balance	Viewpoints, understanding, and competencies must all be in sync to address aspects like demand and capability.
Collaboration	This refers to working together in a cohesive and effective manner. The Kanban system was developed to promote the way in which people work together, therefore collaboration is key.
Customer Focus	Every Kanban system has a goal and has some flow to satisfy its customers by delivering the required service or item. In this situation, customers are external, however, they can be internal to the organization. The value in which customers receive is a central concern in Kanban systems.
Flow	This refers to the realization of work which is a flow that yields value to the organization. This flow can episodic or continuous.
Leadership	This is the ability to inspire others within the organization to reach goals by means of words and reflection. In some organizations, a hierarchical

[8]This section is based on the notes provided by Anderson & Carmichael, (2016). *Essential Kanban Condensed.*

	structure exists, however, Kanban systems require the involvement of all individuals to deliver and improve value to its customers.
Understanding	This refers to the knowledge of individuals to complete certain tasks which are foundational in Kanban because it is an improvement method.
Agreement	This is the shared understanding of the organization's goals and the commitment to achieve them. This can only be done by respecting one another and setting aside individual differences.
Respect	Valuing and understanding the consideration of others. This is the foundation on which all the values are built around.

2.3.3.2 Kanban Implementation Process

The acceptance and adherence of the above Kanban values will make the implementation process of Kanban more efficient. The implementation process can be viewed as seven steps or phases which provide a roadmap to employ Kanban within an organization (Gross & Mcinnis, 2003)[9].

Step 1: Conduct Data Collection

In this phase, all necessary data will be collected to facilitate the production process. The act of data collection requires decision-making, and this data will allow for the calculation of Kanban quantities (will be reviewed in the proceeding step). This step also represents a golden opportunity for conducting value stream mapping for the entire organization to determine the production processes that would be most suitable for the Kanban scheduling systems.

Step 2: Calculate the Kanban Size

The completion of data collection will lead to the calculation of the Kanban size. This involves calculating the Kanban container size based off the current conditions

[9]The Kanban implementation process steps are based-off the descriptions from Gross & Mcinnis (2003). *Kanban Made Simple*. However, it is important to note that several works ascribe to different Kanban practices, but for the purpose of this study only the key activities of Kanban will be reviewed.

and not forecasted plans. These initial calculations use the production requirements, productivity rate, planned downtime, and changes overtime to finalize the replenishment interval.

Step 3: Design the Kanban

The design of Kanban should follow the completion of the calculation of Kanban quantities to support production requirements. This phase kicks off with a visual representation of the workflow from "to-do to done status" (Cole & Scotcher, pg. 71, 2015). The product of this phase should be the implementation of the Kanban, including actions, deliverables, and schedule milestones. Kanban design should be aligned with the organizational culture. This plays a big role in the implementation process to reach common understanding between project members.

The most commonly used design is Kanban cards which were used by the Toyota Production System which relied heavily on the use of cards for signals. However, there are many common Kanban designs used in organizations today, such as Kanban boards depicted in Figure 2.4. Kanban boards are a variation on the Kanban cards as opposed to the use of cards. The board utilizes plastic chips, magnets, etc. This is a useful visual representation and control system for continuous product delivery.

Starting at the Backlog through to "complete," each team needs to define the workflows and states that form the foundation of the lifecycle of their tasks. The visualization component, or cards aids in identifying the state of each task; the readiness, and bottlenecks (Leybourn, 2013).

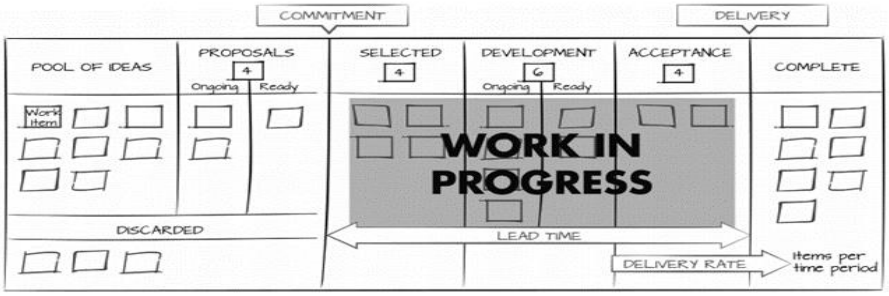


Figure 2.4: An example of a Kanban Board (Anderson & Carmichael, 2016, p. 13)

In addition, the Kanban board indicated above depicts a workflow in which items flow through different stages of a process, in an ordered manner from left to right. The board defines the processes as a series of "knowledge-discovery steps" and their associated policies (Anderson & Carmichael, 2016, p. 13). These policies should be made explicit and go through different stages of work: selected, development and acceptance, respectively.

Step 4: Training

It is imperative before the commencement of Kanban scheduling that the team knows how to work the system as well as their individual role within the process. The training process should include a presentation and review or feedback cycle to ensure the training was successful. Moreover, using real-life cases or scenarios will further enhance their understanding of their roles and decision-making process.

Step 5: Initial Startup of Kanban

Once the setup of the Kanban design and training is completed, the implementation of Kanban scheduling proceeds. For this, to happen all visual management tools must be organized such as signals, control points, and rules. It can be expected after the deployment of Kanban that certain issues will manifest, and the team should be prepared to mitigate any problems that will hinder their progress.

During the development stage, the scheduling transition plan is created to determine the points of change and the required amount of inventory to make these changes. In addition, Kanban limits the number of items to be processed at a given point in time, commonly known as work-in-progress (WIP) – for optimum efficiency (Cole & Scotcher, 2015).

Step 6: Auditing the Kanban

The process of auditing the Kanban follows the initial startup of Kanban. Auditing is the step whereby the auditor monitors the scheduling signals and whether the customers are receiving delivery. Any problems identified from the auditing process must be rectified immediately to maintain the integrity of the Kanban

design. Future requirements will also be reviewed through the auditing process to ensure the Kanban quantities satisfy expected demand. This can be done by managing the flow of work to generate optimum efficiency that will ultimately add business value through consistency and repetition (Cole & Scotcher, 2015).

Step 7: Improve the Kanban

Once the Kanban process is successfully running, improvements to Kanban can be made to minimize inventory quantities, however, the reduction of these quantities should only be made to improve the production process. In order to enhance the process, the team should reduce downtime or changeovers.

The WIP limit is an essential role player in directing the team to focus on blockers when the limit is reached, no more than two tasks per induvial are allowed to avoid inconsistency and problems. The core idea is to improve collaboratively as a team (Cole & Scotcher, 2015). Implementing feedback loops are an essential process control mechanism, thereby improving all areas of the Kanban process such as strategy alignment, risk management, service improvement (Anderson & Carmichael, 2016).

2.3.4 Other Agile Methodologies

The popular agile methodologies as described in the previous sections are used today in many organizations, however, it is also important to acknowledge the existence of less frequently used agile methodologies because often organizations will use an amalgamation of different methodologies that subscribe to the agile manifesto.

Other agile software development methodologies include a variety of methodologies such as Crystal Methodologies, Adaptative Software, Feature Driven Development, Lean Software Development, and Dynamic Systems Development Method. Table 2.5 briefly summarizes these methodologies.

Table 2.5: *Other Agile Methodologies*

Crystal Methodologies	These methodologies are heavily bias towards project management and planning and were developed by Alistair Cockburn in the early 1990s (Mnkandla & Dwolatzky, 2004).
Adaptative Software	The Adaptive Software model is a modified version of the XP model because it uses the prototype approach to verify the design and requirements, hence, the software is developed incrementally as the customer approves the prototype (Qureshi, 2008).
Feature Driven Development	Feature Driven Development was first introduced in 1997 and is a method used in agile development to discover a list of features that will be implemented (Chowdhury & Huda, 2011).
Lean Software Development	Lean Software Development can be described as a set of project management practices rather than a definitive process. It inhibits many characteristics from Scrum (Mnkandla & Dwolatzky, 2004).
Dynamic Systems Development Method	Provides a control framework for rapid development by keeping cost, time at a constant and adjusts the functionality that needs to be developed (Rajagopalan & Mathew, 2016).

In addition, to the above-mentioned agile methodologies, there are also many individual approaches to agile development that do not follow a published

methodology. For the purpose of this study, the review focused primarily on the most popular agile methodologies.

2.4 Extant Reviews on Agile Practices in Remote Working Contexts

The aim of this section is to bring together many of the concepts reviewed in previous sections of this chapter and consider extant research that bears most closely to the objective of this research project. To this end, as stated in chapter 1, this study concerns the implementation of agile practices in remote working contexts. Research in this regard is spread across a number of fields, ranging from Information Systems, Software Engineering, to Project Management and many other related disciplines. To date, three key systematic reviews have been conducted to synthesise extant knowledge on agile practices in remote software development teams (Jalali & Wohlin, 2012; Rizvi et al., 2015; Vallon et al., 2018).

The purpose of and contributions associated with these review papers vary based on their specific methods and research foci, but in general, they focus on the combination of GSD and agile practices. This section aims to highlight how agile software development practices are applied in the context of distributed software development. These three reviews provide the most complete account of agile development practices in distributed software development situations. To this end, a review of each study will be conducted to define and relate specific motivations, contexts and methods followed. This section will conclude with an integrated discussion in which the contributions of these reviews are scrutinised and key gaps in the literature are highlighted.

2.4.1 General Background and Context of the Review Papers

With co-location held up as the “gold standard” for agile software development, given the “new normal” and the continued reality of remote and distributed work, there is a need to understand how agile project management practices can be implemented remotely. Agile practices encourage self-organized co-located teams, compared to distributed software development that implies distribution across cultural, geographical, and temporal boundaries (Jalali & Wohlin, 2012). Therefore,

the combination of agile practices and distributed software development through virtual teams can emerge as a response to address distributed agile development challenges such as continuous collaboration, project costs, timeline constraints, cultural and language issues (Rizvi et al., 2015). Moreover, the distance in GSD implies a different way of working, organizational standards, organizational cultures, and policies, which may decrease the team's cohesion in agile software development because agile methods value face-to-face interaction within co-located teams.

The common theme or context across Jalali and Wohlin (2012), Rizvi et al. (2015), and Vallon et al. (2018) is centered around the idea of incorporating agile in global software development. This ultimately exhibits certain collaboration, coordination, and control challenges that are typically associated with software development projects. In addition, when reviewing Jalali and Wohlin (2012), Rizvi et al. (2015), and Vallon et al. (2018) much overlap of software development project challenges can be observed, such as economic instability, communication barriers, team trust, and cultural issues that appear to be the most prominent in distributed software development literature. In the following sections, each of these papers will be reviewed in the order of publication, commencing with Jalali and Wohlin (2012), then Rizvi et al. (2015), and ending with Vallon et al. (2018). Moreover, each systematic literature review is extensive, and, for this reason, the focus will be on the most relevant aspects of their findings.

2.4.2 Review #1: Jalali and Wohlin (2012)

Motivation and Context

Jalali and Wohlin (2012) argue that there is a host of challenges with GSD and combining agile practices with global or remote contexts yields even greater difficulties. However, despite these challenges, the authors noted that, at the time, adoption of distributed agile development was increasing because of its associated benefits such as shorter time-to-market, reduced software development costs, and mitigating against upcoming challenges. Several challenges can be identified:

distance in GSD requires the alteration of work spatially, and the lack of face-to-face communication among co-located teams may affect team building and trust. Therefore, Jalali and Wohlin (2012) indicated at the time of their review that there was a need to investigate the situation in the research literature to determine how agile practices can be applied in GSD contexts. In addition, Jalali and Wohlin (2012) refer to earlier work by Danait (2005) and Young and Terashima (2008) who found that several software organizations have provided positive feedback on the incorporation of agile practices in distributed software development. These papers further support Jalali and Wohlin (2012)'s review because they indicate the relevance of combining agile practices with remote working contexts.

Jalali and Wohlin (2012)'s aim was to build on an earlier narrative review (Jalali & Wohlin, 2012) and provide a systematic classification and summary of existing research on the adoption and implementation of agile practices in global software engineering contexts. Specifically, this systematic review aimed to identify, firstly, the nature and content of extant research and, secondly, which Agile practices in global software engineering settings are used, and under which circumstances have they been successfully applied.

Scope and Method

Jalali and Wohlin (2012) based the methodology for their systematic review on the guidelines provided by Kitchenham and Charters (2007). The initial phase of the study involved the design of a systematic map following the guidelines provided by Peterson et al. (2008). Jalali and Wohlin (2012)'s search strategy involved, firstly, specifying the Scope that included all agile practices in different types of agile development in peer-reviewed literature (i.e., no grey literature) published between 1999 and 2009 and, secondly, using a series of keyword searches on six databases, (ACM, IEE, Inspec, AIS, Compendex, Scopus) by means of title, abstract and keyword searches. These methods resulted in the inclusion of 81 papers in the analysis.

Findings and Conclusions

Jalali and Wohlin (2012) found that the majority of research at that point was in the form of experience reports, whereby practitioners reported on their individual experiences on a particular issue and methods adopted to deal with them. Most of the experience reports and opinion papers were categorized as qualitative or unclear, and the methodology was either unclear or a case study. Furthermore, the general term "Agile" was commonly used in the literature, and the term "distributed team" was widely used in GSD literature without substantial information. This indicated a lack of contextual and background information in the literature at the time. Ultimately, among the included papers, 63 empirical studies were identified, with 40 written by practitioners, 20 by academic researchers, and 3 were written jointly by practitioners and researchers.

In addition, Jalali and Wohlin (2012) found that 53 success stories were reported in total in the research literature and in most of the successful cases the team was distributed around the globe and worked for long durations on small to medium size projects. Furthermore, the review identified that the most commonly used combination of agile methods and remote working contexts were agile-Offshore, Agile-distributed teams, and XP-distributed teams. In addition to this, the review found that standup meetings were the most frequently used agile practice, followed by sprints/iterations. The least used agile practice was the Scrum master role. Also, the review indicated that collaborations between the USA and India were the most prevalent. Moreover, the review also found distributed development within the USA is also prevalent, and no Asian countries were seen among the customers, however, some Asian countries are popular for outsourcing such as Malaysia and India.

Jalali and Wohlin (2012)'s findings suggest that the application of Agile practices in distributed software development is relatively unexplored. However, the increasing number of studies indicated a growing interest in this domain at the time of the review. For example, in certain cases, agile organizations opted to expand their offices and in others, a distributed organization decided to move to agile as a

result of failure from traditional software development methods. Therefore, Jalali and Wohlin (2012) concluded that agile and global software development in combination have gained significant popularity over the five years preceding the review.

The primary conclusion proposed by Jalali and Wohlin (2012) was that the majority of existing literature was in the form of industrial experience reports in which Agile practices were modified with respect to the context and situational requirements. Moreover, the study observed repetitions in the content of the studies explored. Similar findings were reported in more than one article (e.g., Jalali & Wohlin, 2012 and Vallon et al., 2018). These repetitions indicated that the findings are constant and that there is a consistent pattern of results emerging in literature. Jalali and Wohlin (2012) found that insufficient research and studies analyze the challenges of applying agile in global software development. The challenges and problems are documented in global software development or agile, but the combination was not well examined at the time.

According to Jalali and Wohlin (2012) agile has been successfully adopted within small to medium size distributed projects. The review also found that many studies revealed that agile practices have been customized, and modifications were made to certain agile practices. Furthermore, another type of modification was observed by incorporating different agile practices such as Scrum and Extreme Programming.

2.4.3 Review #2: Rizvi et al. (2015)

Motivation and Context

In a more recent review, Rizvi et al. (2015) indicated that research in distributed software development has rapidly increased in the preceding decade because of the associated benefits such as faster delivery and cheaper labour. These benefits are similar to those described by Jalali and Wohlin (2012). Despite the recent growth of this domain, distributed software development is still evolving. Rizvi et al. (2015) argue that this dynamic and evolving nature of the field suggested the need for a comprehensive systematic review. Furthermore, Rizvi et al. (2015) note that,

despite the benefits brought about by the flexibility of agile methodologies, there exists an incompatibility between the principles of agile software development and those of distributed software development. Therefore, with agile methodologies and distributed team members becoming more prominent they argued that it is was important to understand the challenges faced by software organizations that have implemented distributed agile software development in the past. To this end, Rizvi et al. (2015)'s objective was to understand existing studies of distributed software development and provide solutions to mitigate these challenges.

Rizvi et al. (2015)'s core motivation for their systematic literature review is that, despite the growing number of systematic reviews (i.e., Smite et al.,2010; Hossain et al., 2011) in this domain, software practitioners are still lacking proficiency in this research method, and the number of domains explored is limited. This deficiency is also true in the area of distributed software development; therefore, a need exists for more systematic literature reviews of distributed agile software development. Moreover, they argue that at the time, there were limited systematic literature reviews performed in the specific area of agile practices and remote working contexts.

To point out some of the literature reviews on distributed software development. The paper by Smite et al. (2010) had focused on reviewing empirical evidence in global software development; therefore, it did not focus on agile methodologies. However, the work by Hossain et al. (2011) had a similar theme as Rizvi et al. (2015) because it focused on reviewing the impact and role of Scrum in remote working contexts. However, this was limited to only Scrum and excluded other agile methodologies. The work by Jalali and Wohlin (2012) had the most similarity with the Rizvi et al. (2015) paper as they highlighted the most effectively used agile practices within global software development until 2010.

Scope and Method

The systematic review by Rizvi et al. (2015) mentions that the work by Jalali and Wohlin (2012) has a similar focus with only varying differences in the research

questions of their systematic literature reviews. While the study by Jalali and Wohlin (2012) focused on identifying successful agile practices for global software development, the focus of Rizvi et al. (2015) is more multifold as it focuses on more fundamental research questions that aim to understand, firstly, the reasons and conditions that led to the adoption of distributed agile software development (DASE), secondly, the important risks that can threaten or negatively impact the DASE approach and, thirdly, highlight which available approaches among existing agile methodologies have been most frequently adopted within the software community. The major differentiating factor between Rizvi et al. (2015) and the earlier work of Jalali and Wohlin (2012) is that their focus was to include studies that have a strong experimental, empirical, or case study perspective. For this reason, Rizvi et al. (2015)'s search query had been designed to only include publications in DASE that have an empirical aspect to them. This was not the focus for Jalali and Wohlin (2012) who also included experience reports in their review.

Adopting a methodology similar to Jalali and Wohlin (2012), Rizvi et al. (2015) followed the guidelines provided by Kitchenham and Charters (2002) to conduct their systematic review. To source literature for the review five academic databases (IEEExplore, ScienceDirect, SpringerLink, Wiley Online Library, and ACM) were searched for English-language articles published between 2007 and 2012. While specific eligibility criteria were applied, overall, Rizvi et al. (2015) targeted empirical research papers that either evaluated or implemented a project using agile practices in a distributed context. These search procedures resulted in the inclusion of 63 papers in their review.

Findings and Conclusions

Rizvi et al. (2015) found that the agile methodology that was most frequently adopted in their sample of studies was Scrum which appeared in 40% of the reviewed papers. Furthermore, observations were the most adopted data collection method, and were used in 33 (52%) studies, followed by interviews which appeared in 19 (30%) studies. Rizvi et al. (2015) found that projects in 84% of the studied papers had successfully combined agile into remote working contexts. Rizvi et al.

(2015) indicate that, across the included studies, the main reason to engage with distributed teams was to save cost and outsource talent across the globe.

Rizvi et al. (2015)'s review reveals that the distribution of teams and the time zone differences between the teams are important factors that can show the level of deployment of distributed agile software engineering in practice. The majority of the projects were inclined towards a small number of teams, for example, two or three teams, and peak time zone differences were often restricted to teams that would have some minimum work overlap. This is to alleviate possible coordination and communication issues. It is important to note that this more recent finding supports the earlier findings of Jalali and Wohlin (2012). Further key findings described by Rizvi et al. (2015) indicated that 52% of the projects has engaged in DASE as part of their business practices. These organizations had already engaged with DASE to some level in the past. Moreover, 12% had implemented DASE for experimentation. These were primarily academic projects, and 5% had engaged in agile using distributed teams because of the associated benefits of distributed software development. Lastly, 6% had engaged in agile using distributed teams to simulate real world scenarios.

The operation of agile practices works best with co-located teams, but co-locating is challenging when working with distributed teams, however, it is possible to facilitate co-location through different strategies. One is to allow for distributed team members to start working together at the very beginning, this is known as *seed visits*. An additional strategy would be to allow team members to have contact meetings at different time intervals of the project, this is known as *maintaining visits* or a combination of seed and maintaining visits. Rizvi et al. (2015)'s study found that 16% of the projects distributed teams followed seed visits in the beginning and continued maintaining visits. Furthermore, it was found that 44% of the projects, distributed teams did not collocate, and in 10% of the projects distributed teams collocated in the early phases/iterations. Lastly, 13% of cases distributed teams met during the project through maintaining visits.

Rizvi et al. (2015) refer to numerous risks when adopting distributed agile software development, such risks were categorized into one of four categories: communication, collaboration, coordination, and cultural differences. These four classes of risks in distributed agile software engineering were based on the work performed by Sutherland (2008) and Lee and Yong (2010). In communication many challenges persist such as time zone differences, language differences, lack of synchronous communication infrastructure (i.e., video conferencing or ICT tools) which are all considered one of many risks in distributed agile software engineering. However, more risks present themselves in collaboration where lack of team structure, improper work distribution, and lack of strategic solutions were identified as some challenges. Under coordination, lack of documentation, sharing of sensitive data and costs of synchronous communication were considered major challenges. Lastly, under cultural differences, work practices, holidays, way of speaking, and project timelines were recognized as challenges.

It is important to analyse what mitigation strategies can be used in distributed agile software development. According to Rizvi et al. (2015) in order to deal with communication-related issues, it is imperative to have a solid communication infrastructure, encourage teams to engage in both formal and informal communication, create and enforce communication strategies as well as facilitate regular face-to-face interaction. Furthermore, when dealing with collaboration-related issues it is necessary to facilitate the following: monitor work progress, review lessons learned, planning around holidays, utilizing tools, consistent builds, reduced sprints, smaller teams, decentralize decision-making, and use scrum-of-scrum model. Lastly, many ways present themselves to deal with cultural differences by questioning team members to reach understanding and commonality within the team and interviewing resources before the start of the project are ways in which risks can be mitigated.

Rizvi et al. (2015) conclude that there is still insufficient literature that addresses the failures and successes of projects adopting distributed agile software

development. This finding is also evident in many other studies such as Jalali and Wohlin (2012).

2.4.4 Review #3: Vallon et al. (2018).

Motivation and Context

The third and most recent review into the adoption and implementation of agile practices in distributed software development contexts was published in 2018 by Vallon and colleagues. The paper by Vallon et al. (2018) refers to agile practices as being developed around self-organising and empowered teams with a strong focus on team collaboration and communication that is mediated by agile practices. These agile practices are increasingly found in global software development. The main contribution of their paper is to provide an understanding and analysis of the usage of agile practices within global software development. To this end, Vallon et al. (2018) continued the analysis that was conducted by Jalali and Wohlin, covering the years between 1999-2009 and 2010-2016.

To provide motivation for their study Vallon et al. (2018) argue that global software development is an active research area as several systematic reviews (Marques, Rodrigues, & Conte, 2012; Verner et al., 2012; Raza, MacDonell, & Clear, 2013) all share common opinions in that there is a need for more research for global software development. The most comprehensive and recent review prior to Vallon et al. (2018) was the study conducted by Rizvi et al. (2015) which, as noted previously, investigated distributed agile software development for the years 2007-2012. In addition, as further motivation, Vallon et al. (2018) describe that successful application of agile practices in global software development is very useful to software practitioners. According to the 11th annual state of agile survey report, there is an increased usage of distributed agile teams from 35% in 2012 up to 86% in 2016. Therefore, Vallon et al. (2018)'s motivation was to examine the significant rise from 2012 to 2016 in the implementation of agile distributed teams.

Scope and Method

The scope of Vallon et al. (2018) is focused on the most effectively (state of the art) used agile practices and methods. This is somewhat different from Rizvi et al. (2015) who analyzed the underlying reasons and risks for adopting agile practices within global software development. Therefore, the main differentiating factor between these systematic literature reviews is that Rizvi et al. (2015) aimed to understand agile adoption in GSD, whereas Vallon et al. (2018) is the continuation of the work performed by Jalali and Wohlin (2012), who focused on the global software development domain and the agile practices that have been used in different scenarios. To extend their analysis for the years 1999-2009 to the period 2010-2016, Vallon et al. (2018) chose a similar study design to that of Jalali and Wohlin (2012). Similarly, they also followed the guidelines for systematic review studies by Kitchenham and Charters (2007) and Petersen et al. (2008).

To achieve comparable results to that of Jalali and Wohlin (2012), Vallon et al. (2018) used the same search constraints and terms except ‘open source’ which produced irrelevant studies during pilot searches in certain databases such as IEEE Xplore. The publication years were limited between 2010-2016 and language to English. Furthermore, Vallon et al. (2018)’s search was limited to abstracts, keywords, and titles to minimize the amount irrelevant papers. Their search string looked for agile and global software development, with variants or synonyms of both terms. The databases used by Vallon et al. (2018) were similar to that of Jalali and Wohlin (2012), these databases included: ACM Digital Library, AIS Electronic Library (AISeL), Compendex, IEEE Xplore, INSPEC and Scopus. Moreover, Vallon et al. (2018) excluded studies that, firstly, did not address agile practices in GSD, secondly, were not available via their university library services and, thirdly, were unpublished or grey literature. The final set of 145 included studies for 2010-2016 formed the basis of Vallon et al. (2018)’s analysis. However, in accordance with Jalali and Wohlin (2012), Vallon et al. (2018) selected all successful empirical cases (89 cases out of the 145 studies reviewed) for detailed empirical analysis.

Key Findings and Conclusions

The review conducted by Vallon et al. (2018) found an increase in total publications per year from 82 publications between 1999-2009 to 145 publications between 2010-2016. This indicated a growing interest in the subject of combining agile practices with GSD. From 1999-2009 studies were largely qualitative with 88% classified as qualitative in this period by Jalali and Wohlin (2012). In this more recent review, between 2010 and 2016, 52% of studies were classified as qualitative. Therefore, during 2010-2016 mixed approaches and quantitative approaches were frequently adopted over qualitative approaches, hence, a decline in the usage of purely qualitative methods was found.

In addition, Vallon et al. (2018) identified that case study research was the most frequently adopted research method throughout 1999-2016, with 39% of studies adopting this method. However, in the later period of 2010-2016, there was a rise in literature reviews from 1% in 1999-2009 to 15%. They concluded that this is indicative of the maturing research field. In accordance with Jalali and Wohlin, (2012), Vallon et al. (2018) focused on successful empirical cases within the 145 included studies. For the years 2010-2016, Vallon et al. (2018) included 106 publications with an empirical approach. Out of the 106 studies, 93 cases were extracted and 79 were single case studies and 14 multiple-case studies, of which only 4 reported failures and 89 successes. These 89 cases had successful applications of agile practices in GSD.

As was the case in Jalali and Wohlin (2012), extreme programming was found to be the most frequently adopted agile method in GSD. However, according to Vallon et al. (2018) for the years 2010-2016, no studies solely followed the XP method. However, many cases do make use of XP. Vallon et al. (2018) showed that no cases used XP exclusively, with multiple cases using a combination of both XP and Scrum with Kanban, or lean software development being only applied in the minority of cases. This was also the case for FDD (Feature Driven Development) and DAD (Disciplinary Agile Delivery). Moreover, in 2010-2016 Vallon et al. (2018) found the following agile methods employed in successful GSD cases:

Scrum (53 cases), XP and Scrum (14 cases), unclear agile methods (12 cases), Kanban/lean (8 cases), FDD and DAD (1 each). In addition, the following distribution details were frequently reported: offshore (57 cases), far distance (41 cases), large time difference (30 cases) and insourcing (29 cases). However, cases seldomly reported on contextual details such as team distribution (48 cases), legal entity (38 cases) or geographic and temporal distance (32 each).

According to Vallon et al. (2018) in the period 1999-2016, standup meetings were the most frequently applied agile practice (70.5 cases)[10], hence it was a core agile practice within the GSD environment. This was followed by backlog (52 cases) and sprint/iterations (51.5 cases). The years 2010-2016 followed similar rankings, and the years 1999-2009 also featured standup meetings (18.5 cases) as the most adopted agile practice, followed by sprint/iterations (13.5 cases) and continuous integration (12 cases). In comparison to the years 1999-2009, where six out of ten practices were Scrum-based, in 2010-2016 the Scrum practices were more closely aligned with the implementation of GSD. However, XP practices are less common. This explains why many studies in 2010-2016 used mixed approaches with Scrum and XP.

Based off the findings, Vallon et al. (2018) concluded that the number one most successful distribution scenario is "offshore-Scrum" with many other distribution scenarios being largely Scrum-based: offshore-Scrum (Location): 34 cases; far-Scrum (geographical distance): 28 cases; insourcing-Scrum (legal entity): 20 cases; two Sites-Scrum (number of sites): 20 cases; large-Scrum and Small-Scrum (temporal distance): 20 cases each; integrated Teams-Scrum (team distribution type): 19 cases. In addition, Vallon et al. (2018) concluded that the most successfully used agile practices in global software development across all distribution scenarios are standup meetings, backlogs, and sprint/iterations.

[10]Jalali and Wohlin (2012) refer to half cases in their study, however, no particular explanation was provided for this phenomenon.

Therefore, according to Vallon et al. (2018), Scrum was the most successfully used agile practice in remote working contexts from 1999 to 2016.

The findings and conclusions from Vallon et al. (2018) indicate that eight of the top ten agile practices used in global software development from 1999-2016 are core Scrum practices integrated with XP practices, hence the term "continuous integration" as the 8th most frequent and pair programming sitting at 10th. The findings support that agile global software development is a maturing research field and that there is an average of 20 publications per year in the field since 2008.

2.4.5 Concluding Remarks of Review Papers

This section aims to highlight the core findings and conclusions of the three literature reviews. Jalali and Wohlin (2012) found that several practices have been applied in distributed software organizations such as stand-up meetings, sprint/iterations, continuous integration, sprint planning, retrospectives, pair programming, sprint review/demo, test-driven development, Scrum of Scrum, onsite/proxy customer, and backlogs. During the course of their study, they observed that researchers and practitioners have different perceptions of the nature of agile practices. Therefore, there is a need for practitioners to collaborate closely and illustrate the practices.

The systematic literature review by Rizvi et al. (2015) was based on 63 selected primary studies that were closely aligned with the research questions. The study found that 35% of these papers evaluated a method, tool, practice, or framework while 65% of them captured the risks/mitigations. To further breakdown, 25% of the papers evaluated a framework, tool, or practices within an industry setting. Furthermore, Rizvi et al. (2015) literature review revealed that time zone difference, knowledge of resources, lack of infrastructure, responsibilities, and missing roles to be prominent challenges facing distributed agile software development. In their study, it was noted that there is a need for tracking and documentation of success and failure of distributed agile software engineering projects.

The works by Vallon et al. (2018) were closely related to that of Jalali and Wohlin (2012) as they reviewed the usage of agile practices in global software development for the years 1999-2016 as well as compared the studied period of Jalali and Wohlin (2012). Vallon et al. (2018) found a steady research output trend of approximately 20 publications per year since 2008, most of which displayed higher quality contributions compared to 1999-2009. There is a shift from experience reports, which were predominant up to 2009, to evaluation studies in 2010-2016. This pattern is associated with the application of more quantitative research methods compared to qualitative case study research. Furthermore, the study revealed several modifications and extensions of agile practices to mitigate against the challenges of global software development. Therefore, despite XP being the predominant agile practice during 1999 to 2009, a predominance of Scrum, often the combination of Scrum and XP was evident (Jalali & Wohlin, 2012).

Following the analysis conducted by Vallon et al. (2018) and given the 'new normal' and continued reality of remote work leading up to and during the covid-19 pandemic, there is a need to perform a more recent evaluation of the usage of agile practices in remote working contexts. It can be observed through the conceptual analysis of Jalali and Wohlin (2012), Rizvi et al. (2015), and Vallon et al. (2018) that much overlap exists in terms of their motivation and context that being a need for systematic literature reviews within the global software development environment. I would also argue that 1) the rapid transition to remote work brought about by the pandemic, 2) the research focus on this phenomenon before and during the pandemic, and 3) the time since the previous review imply a need for an updated synthesis of existing knowledge on the implementation of agile practices in remote working contexts.

In order to continue research within this domain numerous future directions are needed to improve the state of the art in the field of agile GSD. Vallon et al. (2018) identified several futures directions, firstly, the need to report complete empirical context in future studies, secondly, the need for more research covering specific knowledge areas and, thirdly, there are opportunities to investigate more core agile

development practices in GSD and report on more failure cases on applying agile practices in GSD. It is important to note that some reviews are built around existing works, and this can affect the quality of the findings. Furthermore, it is difficult to measure the accuracy of the findings from existing studies, due to assumptions made. Also, when following a systematic procedure, it is difficult to identify whether other researchers have used similar studies and if they had a slightly different set of results and trends.

2.5 Conclusion

The literature review component of this thesis has defined key concepts of both remote work and agile software development. The agile development section provided a high-level overview of software engineering, described the core values of agile development, and discussed the commonly used agile methodologies. In addition, three systematic reviews were analyzed in order of publication. The most important conclusion made after analyzing the systematic literature reviews is that there is a need to perform a synthesis of more recent research and identify knowledge gaps within distributed software development. Therefore, the central focus of the thesis is the successful application of agile practices within remote working contexts, and how this has accelerated since the beginning of the pandemic.

Chapter 3

Research Methodology

This project aims to synthesise current knowledge on the implementation of agile in remote working scenarios in the context of software development teams. To this end, the investigation concerns the practices, tools, roles, and challenges currently described in the literature. Vallon et al. (2018) conducted a systematic review for the years 2010-2016 to continue the analysis of Jalali and Wohlin (2012). Therefore, to provide a more recent review and capture changes during the Covid-19 pandemic, this study aims to expand on the work of Vallon et al. (2018) by continuing their analysis and providing updated results for the period since their analysis. To continue the work of Vallon et al. (2018), and compare findings, this thesis follows a similar research methodology to that of Vallon et al. (2018) with a few alterations which will be explained in the latter part of this chapter. The adopted research methodology will be mixed with both qualitative and quantitative components. Additionally, this chapter provides details surrounding the review protocol employed to guide the conduct of this research and adheres to the guidelines for systematic review study design by Kitchenham and Charters (2007). The general principles and assumptions of systematic reviews is they start by defining a review protocol that specifies the research question being investigated and the methods used to perform the review. Additionally, systematic reviews are based on a defined search strategy to detect relevant literature through the use of inclusion and exclusion criteria. Systematic reviews are a prerequisite for quantitative meta-analysis (Kitchenham and Charters, 2007).

3.1 Research Questions

In order to update and continue the analysis of Vallon et al. (2018) similar research questions are used. Specifically, RQ1a, RQ2a and RQ3 aim to provide updated results and new interpretations. Moreover, this thesis intends to compare the results of the newly studied period of 2017-2021 to the former research period 2010-2016

as well as Jalali and Wohlin (1999-2009) period, by adding RQ1b and RQ2b to analyse the possible changes that may have occurred in the recent years. The research objectives have been translated into five specific research questions as follows:

- ***RQ1a:*** *What is reported in peer-reviewed literature about agile practices in software development in remote working scenarios between 2017 and 2021?*
- ***RQ1b:*** *Compared to the period 1999 - 2016, what reporting differences exist in the peer-reviewed literature about agile practices in remote working scenarios for the period 2017-2021?*
- ***RQ2a:*** *Which agile practices, in which remote working settings, under which circumstances have been successfully applied in peer-reviewed research literature between 2017 and 2021?*
- ***RQ2b:*** *How do the results differ for the period 2017-2021 compared to those of 1999-2016?*
- ***RQ3:*** *Which agile methods, in which remote working settings have been successfully applied in distributed software development, and why?*

RQ1a concerns the implementation of agile practices in remote working contexts in recent times. Extending this RQ1b has been posed to guide the investigation of any significant changes of results in recent times compared to earlier periods. In addition, RQ2a helped understand how to implement agile practices successfully in remote working contexts in recent times. Following this RQ2b has been formulated to gauge any possible changes of results in the successful implementation of agile practices in different scenarios in recent times. Lastly, RQ3 has been posed to increase the knowledge on the application of agile practices in distributed software development to understand under what circumstances have cases failed.

The core motivation for this research is to analyse the significant rise from 2017-2021 in the implementation of agile distributed teams in the software development industry today (Henry, le Roux & Parry, 2021). Therefore, it is important to evaluate whether these recent years produce a different set of results since 2016.

Hence, RQ1b and RQ2b directly link to that objective, while RQ1a and RQ2a address the goal of providing a complete and updated state of the art overview of 2017-2021 by continuing the work of Vallon et al. (2018). In addition to this, RQ3 addresses an area of research that was not extensively studied by Vallon et al. (2018) due to the lack of failure cases reported.

3.2 Systematic Review Design

Following the recommendations by Kitchenham and Charters (2007), the research method adopted in this thesis involved defining research questions, reviewing the scope, conducting searches on data sources, searching papers, reviewing abstracts, reviewing the classification scheme, extracting data to answer the research questions, and documenting the results. These phases are illustrated in Figure 3.1. The planning phase of the review involved specifying research questions and developing a review protocol outlined in this chapter. Furthermore, conducting the review entailed identifying portals for paper searches, the search query definition, filtering search results using eligibility criteria to further select appropriate papers, and the data extraction and synthesis process. While conducting the review, several primary studies are identified and selected based on the search query followed by the quality-based criteria on the results, and data extraction process. During the reporting of results, data is analysed, and results are summarized. The results and analysis will be presented in Chapter 4.

Figure 3.1: Systematic review phases

According to Figure 3.1, the initial step involved the identification of relevant research questions to answer any possible changes within the agile distributed software development domain in recent years. Furthermore, in order to address the research questions, two data types are described in Figure 3.1 which is existing data that is extracted (summary data reported in Vallon) that comes from Vallon et al. (2018) and then Jalali and Wohlin (2012), and new data for the period 2017 – 2021. Extending this, the review scope was designed to specifically focus on agile practices within remote working contexts in order to update Vallon et al. (2018)'s study. Following the identification of the research questions and scope the study involved the formulation of various search strings for each database (see Section 3.3.2) to obtain a collection of studies that was related to the research questions. Furthermore, the classification scheme was based on the title, abstract, keywords for each study to conclude with a final set of relevant studies for the data extraction and analysis procedures.

3.3 Data Sources and Search Strategy

In the sections which follow outline the details for data collection for both data sources and search strategy.

3.3.1 Eligibility Criteria

For the collection of new data, only studies published between 2017 and the time of the review (August 2021) were eligible. To be included studies must 1) directly link to the research questions; 2) address agile practices in remote working contexts in software development; and 3) the published study report must be available on the University library services during the time of research (Stellenbosch University) or freely available on the web. In accordance with Vallon et al. (2018), studies that were potentially not peer reviewed (i.e., theses, books, workshop papers) were excluded from the review. Additionally, papers published in languages other than English as well as other forms of grey literature were also excluded. The selection process will be illustrated in the next chapter in a PRISMA flow diagram.

The full eligibility criteria are shown in Table 3.1. Vallon et al. (2018) stated that their exclusion criteria are aligned with Jalali and Wohlin (2012), and the criteria extracted from their report did not include studies without results and studies that have been published multiple times, which is stated as criterion 9 in Table 3.1. For conducting this research, modified versions of Vallon et al. (2018)'s criteria for quality assessment have been included and additional inclusion and exclusion criteria have been outlined. The criteria are similar to those used by Vallon et al. (2018) who adopted a knock-out rule (e.g., if one criterion was not met, the study is regarded as unfit to be included for review).

Table 3.1: Inclusion and Exclusion criteria for studies

Inclusion Criteria	
1	Papers that address agile practice in remote working contexts
2	Papers that refer to specific agile practices not as a whole context
3	Papers published between 2017-2021
4	Papers are written in English
5	Papers where search terms found in the title and/or abstract
6	Peer-reviewed papers
7	Papers where the methodology of the study achieves the objective
8	Papers where the results from the study are accurately reported
9	Papers that report on original results that have not been published elsewhere
Exclusion Criteria	
1	Papers that are duplicates of included studies
2	Papers that address agile practices without remote or distributed teams
3	Papers that focus on remote work without agile practices

Criterion 1 ensures that the study focuses on agile methods, (e.g., Scrum, Extreme Programming, or Kanban) in remote working environments and has a suitable context. In addition, criterion 2 demands that only studies describing concrete agile practices have been included. Therefore, the combination of criteria 1 and 2 ensures

that the selected studies address which agile methods have been successfully used in distributed environments as well as failure cases. Furthermore, to address inclusion criterion 9 (i.e., multiple publications of the same results with minor variations), only the first study published will be included for review. There was no additional quality assessment for the years 1999-2016 (i.e., the studies included by Vallon et al. (2018) and Jalali and Wohlin (2012)).

3.3.2 Search Strategy

The process of identifying relevant papers in the field of distributed agile software development was performed on all digital libraries used by Vallon et al. (2018) except for AIS Library and INSPEC. During the forward and backward procedures this can, to some extent, cover this limitation. However, SCOPUS does cover the majority of articles published in AIS Library and INSPEC. The available databases on the Stellenbosch University Library were namely: IEEExplore, ACM digital library, Compendex and Scopus. These digital libraries have been consistently used by Jalali and Wohlin (2012) and Vallon et al. (2018), and others as well in the past for performing literature reviews in software engineering. In order to achieve comparable results to Vallon et al. (2018), similar search terms and constraints were used, however, search terms were expanded to account for newer terms (i.e., often the term “distributed” would be replaced with the term “remote”). The publication years were set to 2017-2021 and the language to English. The database search strategy was limited to abstract, keywords and title to minimize the number of irrelevant hits. In addition, books were excluded from database searches as they generally are not peer-reviewed. Following Vallon et al. (2018), the search string looked for agile and distributed software development, with variants and synonyms of both terms separated by OR-operators, as follows:

> (agile OR scrum OR "extreme programming" OR "pair programming" OR "lean development" OR "lean software development") AND ("global distributed 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 "remote team" OR "virtual team" OR offshore OR outsource)

It is important to note that the aforementioned search string is a generic search string as in many cases alterations were needed for each database. Appendix A provides the specific terms and constraints for each database along with the number of results returned by each. Moreover, the data extraction process was conducted based on title, keywords and abstract then followed by a full-text analysis to obtain a final set of studies. In addition to this search procedure, in accordance with Webster and Watson (2002), *backward* (reviewing citations of articles) and *forward* (using Web of Science) were used to expand the search results and identify articles citing or cited by the relevant literature for the review. These procedures were applied to the final set of eligible studies based on the electronic search process.

3.4 Data Extraction and Management

Data from the included studies (i.e., both the newly identified studies and those included in Vallon et al., 2018 and Jalali and Wohlin (2012)) were extracted and managed in a spreadsheet for subsequent analysis. The data extraction plan and layout are closely related to that of Vallon et al. (2018), whereby identical data points are extracted from each study. The data extraction form, shown in Appendix B, was designed to accrue all the required information to address the research questions and quality assessment criteria for studies on agile distributed software development. In addition to acquiring all the needed data to address the research questions and quality assessment criteria, the following data were also extracted from each primary study as *General* (e.g., the title of study, authors information, publication target and year), *Research* (method, sub-method, type and means of analysis) and *Results* (type of contribution). For the number of successful cases among the included studied papers, featured in experience reports or case studies, the following information was also extracted: *Empirical* (i.e., project duration and size), *Distribution* (i.e., location) and *Agile* (i.e., agile methodologies, agile practices, and agility level). The purpose of collecting the aforementioned information is to provide a full-text analysis of the studies themselves.

3.5 Data Analysis

The extracted data were represented in a spreadsheet for data synthesis. In addition, the exisiting data from Vallon et al. (2018) and then Jalali and Wohlin (2012), as extracted from their literature review, was added to the same spreadsheet. Like Vallon et al. (2018) all the included studies for data analysis were classified into one of the following research groups:

- **Solution Proposal:** proposal of new solution technique without substantial evidence but offers an example or proof of concept.
- **Validation Research:** Investigation of the properties of a solution that has not been implemented in practice, methods may include (i.e. experiments, prototyping).
- **Evaluation Research:** Analysis of a problem or the implementation of a technique in practice by means of (i.e. case study, survey).
- **Philosophical studies:** A new conceptual framework.
- **Opinion paper:** Statement of the authors personal opinion.
- **Experience paper**: Listing of authors personal experience**,** also written by industry practitioners.

Data synthesis was divided into quantitative and qualitative synthesis (this will be presented in more detail in the upcoming chapter). Synthesis for multiple cases was conducted by considering each case study separately. The quantitative analysis was multifold and, similarly to Vallon et al. (2018), all included studies were examined by identifying the number of successful empirical cases and their characteristics as well as reporting on the number of failure cases. The quantitative analysis aimed to answer the defined research questions. Following this, the qualitative analysis addresses a holistic view of 2017-2021 (RQ1a, RQ2a) as a continuation of Vallon et al. (2018)'s study as well as a comparison of the newer studied period 2017-2021 against the former results of 1999-2016 (RQ1b, RQ2b). Furthermore, for RQ3, addresses the number of cases that successfully applied agile methods in GSD. Successful cases were classified as successful by default unless the study mentioned the unsuccessful use of agile practices, it then would be considered as a failure case.

Vallon et al. (2018) included the entire time period between 1999 and 2016 for their data analysis, however, they only collected new data from 2010-2016. In order to compare to the earlier period, Vallon et al. (2018) extracted the data from the earlier period directly from Jalali and Wohlin (2012). Therefore, in order to conduct a full comparison of 1999-2021 (inclusive of both previous periods and the new period) were examined. To address the research questions, the data will only be extracted from Vallon et al. (2018) and then Jalali and Wohin (2012) in addition to more recent work. Therefore, the data analysis will examine two types of data: "new data" produced through the systematic search process (2017-2021) and "existing data" extracted from Vallon et al. (2018) and Jalali and Wohlin (2012) that is all of their included studies (1999-2016).

For this study, all geographically distributed development teams were included, not only globally distributed teams. This aligns with Vallon et al. (2018)'s study. Moreover, this thesis follows the expanded classification used by Vallon et al. (2018) for remote working types/scenarios to incorporate the distributed software development taxonomy with the following definitions:

- **Location:** Offshore (different countries) > Onshore (same country) or unclear
- **Legal Entity:** Outsourcing (different companies) > Insourcing (same company) or Unclear.
- **Geographic distance:** Far (flying time 2 h and more) > Near (flying time less than 2 h) or Unclear.
- **Temporal distance:** Large (more than 4 h) > Small (4 h or less) or Unclear.

Vallon et al. (2018) used the sign ">" to denote any study that falls into several distribution types. To stay closer to the taxonomy in this study similar data to Vallon et al. (2018) were extracted in order to compare results. The overall project size is also defined as in Vallon et al. (2018) with: Large (more than 50 persons) > Medium (21–50 persons) > Small (20 or less persons) or Unclear. Project duration was defined as: Long (more than 7 months) > Medium > Short (less than 1 month) or Unclear. Moreover, in alignment with Vallon et al. (2018) the term *agile practice*

originates from an agile method as described before (such as Scrum, Extreme Programming, Kanban). From the full-text analysis, a study was considered a success if the application of agile practices was successful in that software development requirements were met. (2018). Similarly, to Vallon et al. (2018), this study considers all applications of agile practices within a study a success if the study has a successful overall outcome and all applications of agile practices as failed if the study was not successful overall.

3.6 Chapter Synopsis

This chapter provided the methodological details for the procedures employed to guide the conduct of the review. It discussed the systematic review design, data sources and search strategy, eligibility criteria, quality assessment criteria, data extraction procedures, and data synthesis procedures. Moreover, many variables and methods followed were similar to that of Vallon et al. (2018) in order to avoid compromising the validity of the study because the aim is to continue the research of Vallon et al. (2018)'s study. Hence, in order to continue the works of Vallon et al. (2018) a more recent review i.e., 2017-2021 was deemed necessary, especially given the changes brought about by the Covid-19 pandemic. Furthermore, the systematic review design guidelines followed in this study was inspired by Kitchenham and Charter (2007), these steps are depicted in Figure 3.1, and because the systematic review approach provided a wider perspective across multiple studies compared to studying a single entity within an organization. Similar research questions were used in order to update Vallon et al. (2018) as the core motivation of this section is to analyse the significant rise from 2017-2021 in the implementation of agile distributed/remote teams in today's industry compared to 2010-2016.

To find relevant papers and stay as close as possible to the study by Vallon et al. (2018), the following digital libraries were used: IEEExplore, ACM digital library, Compendex and Scopus. The inclusion criteria were defined to include studies published between 2017-2021 that directly link to the research questions. Furthermore, in order to achieve comparable results to Vallon et al. (2018)'s study,

the same search terms and constraints were used. The data extraction process was similar to that of Vallon et al. (2018) as the data extraction process was conducted based on title, keywords and abstract then followed by a full-text analysis of studies.

Chapter 4

Analysis and Results

This chapter describes the study inclusion process and reports the results of the comparative and qualitative analyses of the selected studies. To this end, the systematic literature search resulted in 148 included studies for the period January 2017 to September 2021 to be used for the comparative (Section 4.2), quantitative (Section 4.3), and qualitative (Section 4.4) analyses. The comparative analysis will provide insight and comparison between the existing data (i.e., Jalali & Wohlin, 2012; Vallon et al., 2018) and the newly collected data. Extending this general analysis, as was the case in Vallon et al. (2018), the qualitative analysis will only consider empirical cases and their characteristics within the set of the 148 included studies.

4.1 Study Inclusion Process

As described in the previous chapter, a step-by-step study inclusion process was followed, whereby all databases were searched with pre-defined search strings to identify any potentially relevant papers. The search strategies produced a total of 460 results. After duplicates (n = 204) were discarded the titles and abstracts of the remaining unique results (n = 256) were screened. Ineligible records (n = 111) were removed before the full texts of the remaining studies (n = 145) were considered. This process was conducted by the primary author, with a separate assessment from a second reviewer. Records that were agreed to be ineligible based on the stated study inclusion criteria (n = 14) were removed. Additionally, as their full-texts were unavailable six further papers were removed (See appendix C for full references of these studies). These screening procedures produced a sample (n = 125) with which backward and forward searches were conducted. The final sample, supplemented by the results of these searches (n = 23), was then established (n = 148). This process is summarized in a PRISMA flowchart depicted in Figure 4.1.

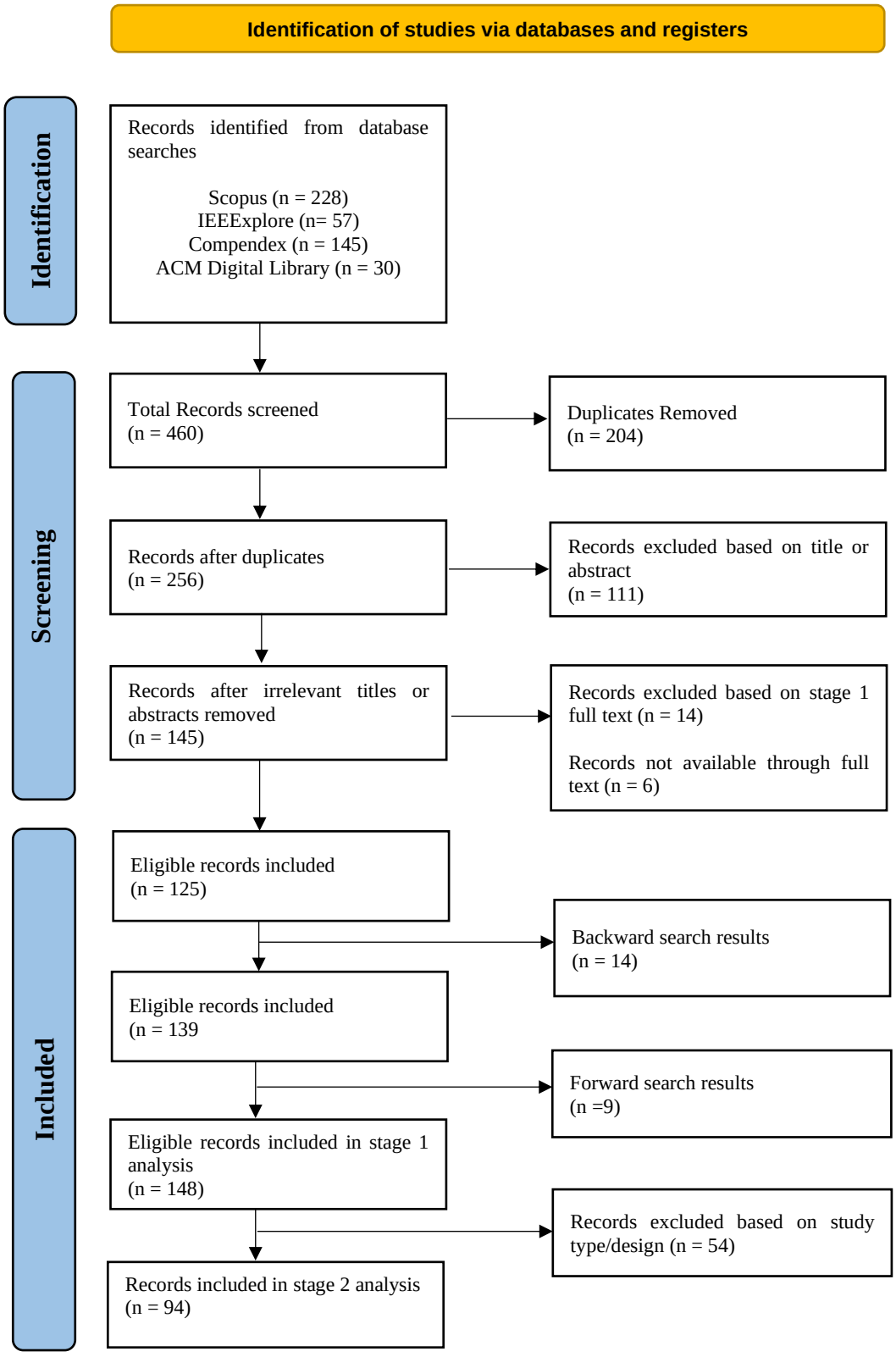


Figure 4.1: A PRISMA flowchart for study inclusion.

In accordance with Jalali and Wohlin (2012) and Vallon et al. (2018), the stage 2 analysis, which will be presented in Section 4.3 after the comparative analysis, will involve only the empirical cases within the 148 included studies. For the years 2017-2021, 94 publications adopted one of the following empirical designs for data collection: case study, multiple-case study, interview, observation, questionnaire/survey, action research, experiment, simulation, and quasi experiment. Therefore, the remaining 54 publications (i.e. literature reviews and unclear studies) were omitted from further quantitative analysis. Moreover, an additional 10 cases were extracted from the multiple-case studies. This made a total of 104 empirical studies, out of which six were classified as failures (see criteria in the previous chapter) and 98 as successes.

4.2 Comparative Analysis

This section compares and contrasts the "existing" (1999-2016) and "new" data (2017-2021), and reports on any substantial changes that may have occurred during 2017-2021. To achieve this, five characteristics are considered each with a unique set of data and objective, namely: distribution of publication types, distribution of research methods, distribution of research sub-methods, distribution of means of analysis, and distribution of study contribution.

4.2.1 Mapping of publication types

Figure 4.2 depicts the mapping of research publication types for the entire period of 1999-2021 which includes data from Jalali and Wohlin (2012) and Vallon et al. (2018). In 1999-2001, as reported in Jalali and Wohlin (2012), no publications on agile practices in GSD were identified. Based on the illustration it is evident that experience reports were the most frequent publication types by far from 1999-2009 at 42 publications. However, according to Vallon et al. (2018)'s data, it is clear that the maturing research field shifted from experience reports to evaluation studies in 2010-2016 with 79 publications being evaluation-based studies. Furthermore, the newly collected data shows that evaluation studies remain the most dominant publication type for the years 2017-2021 at 110 publications. Moreover, evaluation

studies appear in 21% of cases in 2010-2017 compared to 29% of cases in 2017-2021. Therefore, it can be noted that evaluation studies have increased by 8% for the years 2017-2021.

As is evident in Figure 4.2, a majority of research into agile methods in GSD takes the form of evaluation studies (54%). In evaluation studies, the reseachers analyze a problem or the implementation of a technique in practice (e.g. field study or a survey and case studies). Compared to the remaining categories (Solution: 8%, Experience: 20% and Validation: 7%), it is clear that this type of research dominates the field. In contrast, only 13 opinion papers (3%) and 26 more philosophical papers (7%) have been published. It is important to note that some categories i.e. evaluation studies have increased considerably (note that 2021 is incomplete).

Figure 4.2 represents additional publication types that are less dominant compared to evaluation studies and experience reports. It is important to note in Figure 4.2 that 6% of publications were validation studies for the years 1999-2009 compared to 7% publications in 2010-2016, and 9% publications in 2017-2021. This can be indicative of a growing trend due to the increasing amount of published validation studies over the years 1999-2021. Moreover, it is evident that opinion, philosophical and solution studies are relatively uncommon publication types within the GSD research domain. However, according to Vallon et al. (2018)'s data there were 19 publications for philosophical studies for the years 2010-2016 which is substantially larger compared to 2 publications in 1999-2009 and 5 publications during 2017-2021. Furthermore, the cumulative number of publication types for the full period 1999-2021 follows: 74 (20%) publications were experience reports, 13 (3%) publications were opinion-based studies, 26 (7%) publications were philosophical studies, the most common was evaluation studies at 203 (54%) publications, while 28 (7%) publications were validation studies, and finally 31 (8%) publications were solution-based studies.

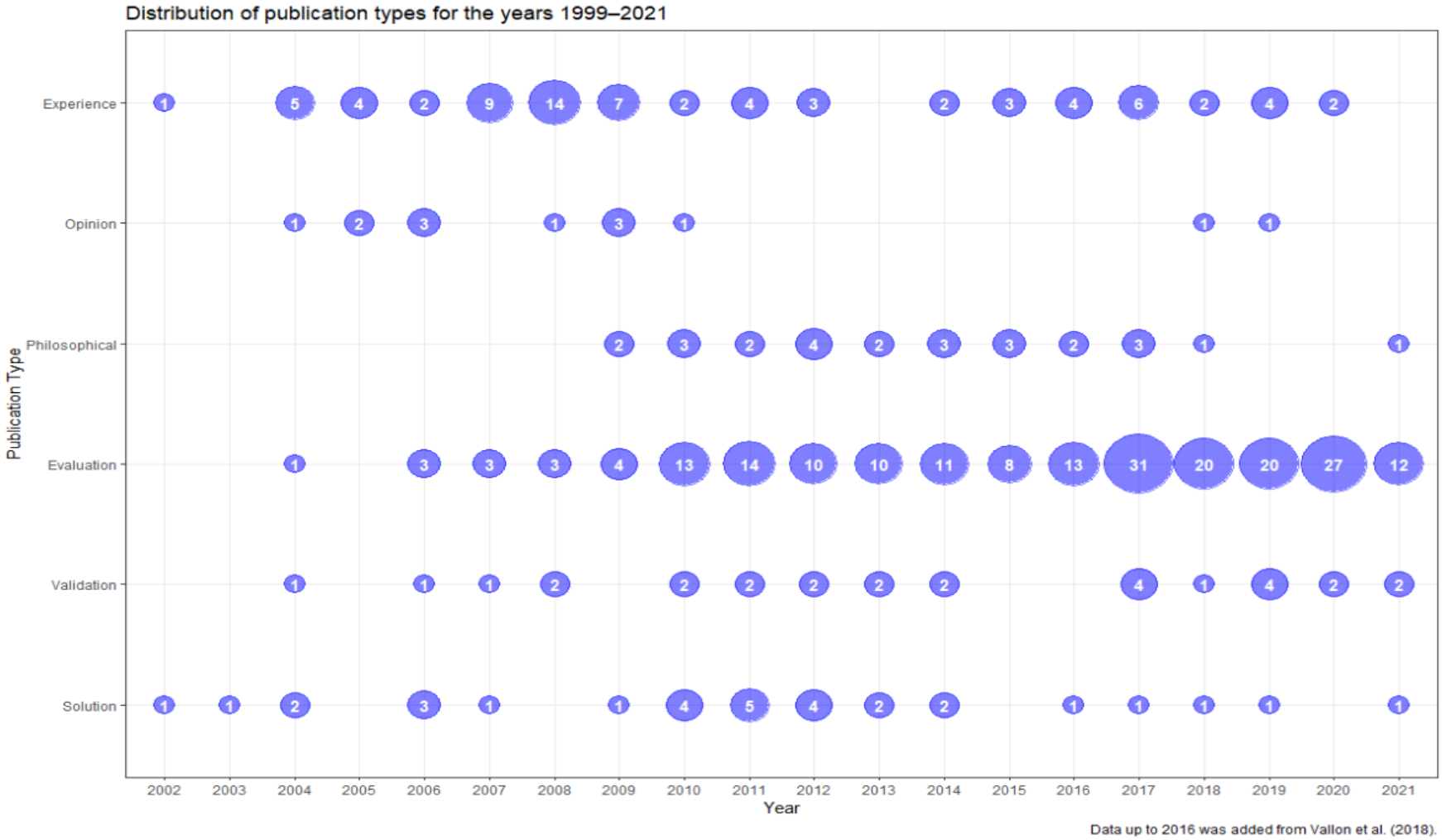


Figure 4.2 Distribution of publication types over the years 1999-2021.

4.2.2 Distribution of research methods

This section compares the distribution of research methods for the full period 1999-2021. As depicted in Figure 4.3, it is clear that, across the last twenty-two years, qualitative methods are dominant, with 88% in 1999-2009, 52% in 2010-2016, and 40% in 2017-2021. However, the proportion of studies adopting qualitative methods has declined over the period. This is matched by the growth in studies adopting mixed method approaches. In 2017-2021 it was found that mixed approaches were used in 33% of the studies as compared to 22% in 2010-2016 and 4% in 1999-2009. This is indicative of a growing adoption of mixed methods in studies within this research domain. Furthermore, quantitative approaches were used in 9% of the studies for the years 2017-2021 as compared to 13% in 2010-2016 and 2% in 1999-2009. This indicates a slight decrease in the number of studies using purely quantitative methods in 2017-2021 compared to 2010-2016, but still more than the early years.

The decrease in the usage of purely qualitative methods for the years 1999-2021 is a result in favour of more mixed and quantitative approaches. In addition, the number of studies that used agile methods that were unclear has increased to 18% in 2017-2021 compared to 13% in 2010-2016 and 6% in 1999-2009. Therefore, based on Figure 4.3 comparing the “existing data” and “new data”, qualitative methods remain largely dominant in studies. However, mixed approaches are steadily on the rise, hence, the adoption of more mixed approaches may exist in future studies.

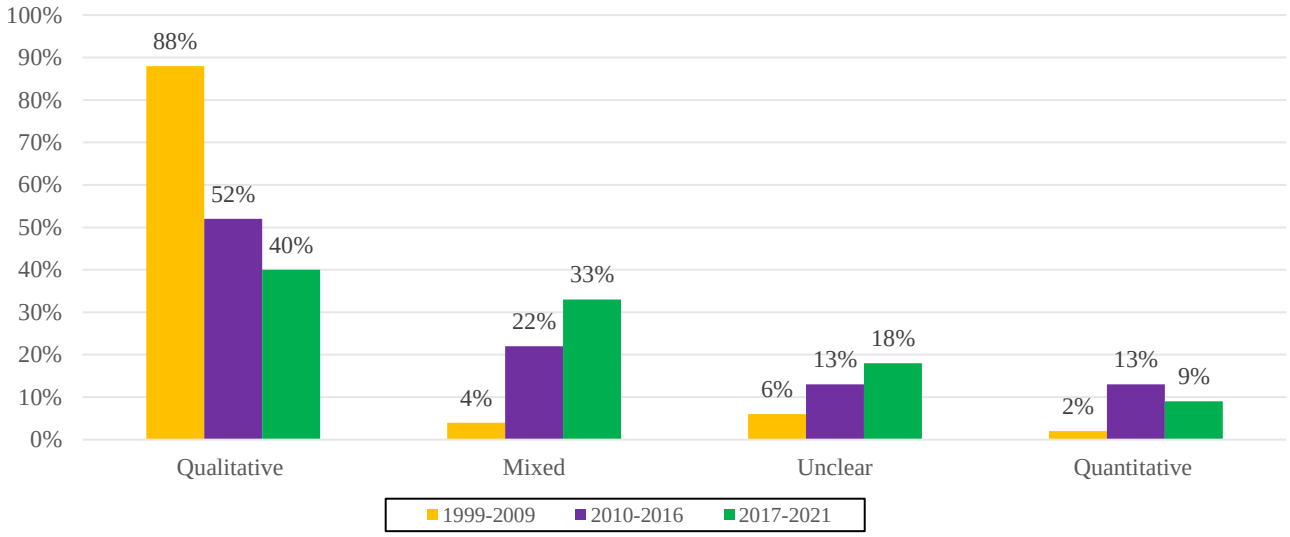


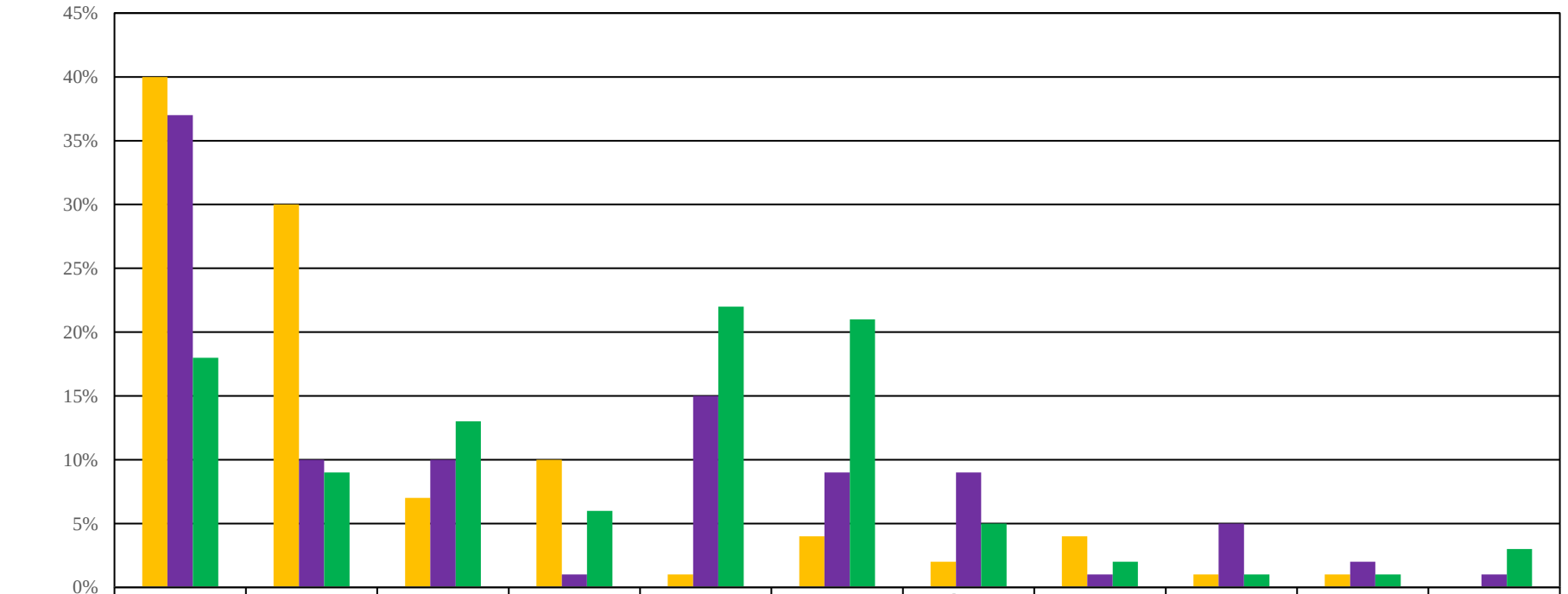
Figure 4.3 Distribution of research methods for 1999-2021, data up to 2016 was added from Vallon et al. (2018).

4.2.3 Distribution of research sub-methods

This section compares the distribution of research sub-methods for the full period 1999-2021. In relation to Figure 4.4, case study research was the most frequently adopted research sub-method throughout 1999-2016 with 40% in 1999-2009 and 37% in 2010-2016. However, the new data reveal that only 18% of studies in 2017-2021 used case studies which is significantly less compared to the previous years. However, the middle period of 2010-2016 showed that there was a rise in literature reviews to 15% compared to 1% in 1999-2009. This rise has continued to escalate as the new data show that 22% of studies are literature reviews which is the most frequently adopted research sub-method for the period 2017-2021. This can be explained by a maturing research field and is a result of the decreased usage of case studies for the years 2017-2021.

In 2017-2021 it was found that 21% of studies used either a questionnaire or survey or both as their research sub-method compared to 9% in 2010-2016 and 4% in 1999-2009. This also explains the decreased usage of case studies in favour of more cross-sectional, survey-based methods. Furthermore, the usage of interviews has also increased to 13% in 2017-2021 compared to 10% in 2010-2016 and 7% in 1999-2009. In addition to this, there are also fewer studies with an unclear approach from

2017-2021, which also supports the observation of a maturing field. Finally, based on Figure 4.4 it can be found that research sub-methods such as observations, multi-case studies, experiments, simulation, quasi-experiment, and action research all make up a relatively small portion of distributed research sub-methods of the 148 included studies. Therefore, it is evident that the above-mentioned research sub-methods are not widely used amongst studies.



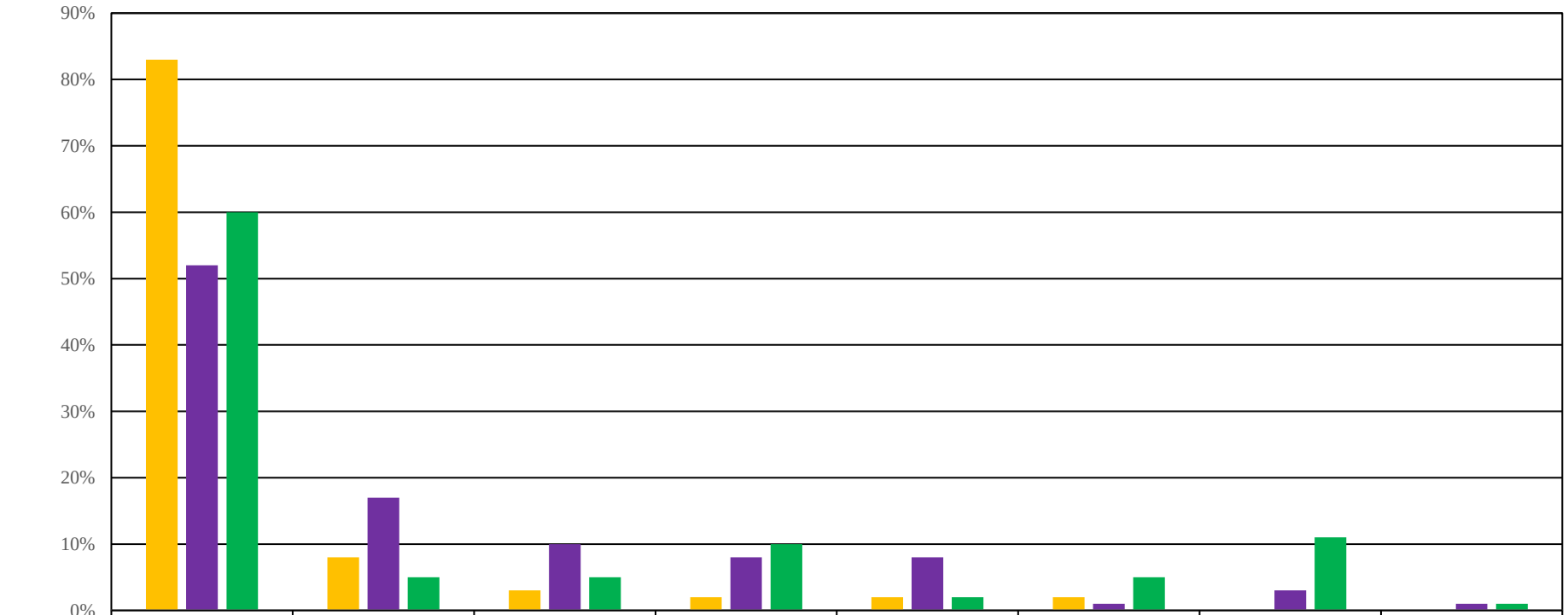
	Case Study	Unclear	Interview	Observation	Literature Review	Questionaire/Survey	Multi-Case Study	Experiement	Simulation	Quasi-Experiment	Action Research
1999-2009	40%	30%	7%	10%	1%	4%	2%	4%	1%	1%	0%
2010-2016	37%	10%	10%	1%	15%	9%	9%	1%	5%	2%	1%
2017-2021	18%	9%	13%	6%	22%	21%	5%	2%	1%	1%	3%

Figure 4.4. Distribution of research sub-methods for 1999-2021, data up to 2016 was added from Vallon et al. (2018).

4.2.4 Distribution of means of analysis

This section compares the distribution of means of analysis for the full period 1999-2021. Figure 4.5 shows that descriptive means of analysis were the most frequently used from 1999 to 2021. The usage of descriptive means of analysis was at 83% from 1999 to 2009 but has decreased to 52% in 2010-2016. This may be a result in favour of other means that align with observations regarding research methods in general (Figure 4.3) that implies qualitative approaches are used less in the later period. Also, the rise of mixed methods studies has influenced this fluctuation. However, according to Figure 4.5, the usage of descriptive means of analysis has increased to 60% in 2017-2021. This increase may be due to the lack of preference or usage of other means of analysis such as comparison that was at 17% in 2010-2016 and then decreased to 5% in 2017-2021. This was also evident with grounded theory which was at 10% in 2010-2016 and decreased to 5% in 2017-2021. This is due to the decrease in qualitative methods.

Despite the noticeable decrease in usage of descriptive means of analysis during 2010-2016, descriptive means of analysis remains the dominant means and is the most widely used in studies compared to its counterparts. Also, it was found that statistical means increased to 10% in 2017-2021 as compared to 8% in 2010-2016 and 2% in 1999-2009. Another observation is that the number of studies that used other means of analysis has increased to 11% in 2017-2021 compared to 3% in 2010-2016 and none in 1999-2009. Also, it is important to note that more studies used tool development as means of analysis with 5% in 2017-2021 as compared to 1% in 2010-2016 and 2% in 1999-2009. Furthermore, it was found that fewer studies used measurement as a means with 2% in 2017-2021 which was identical for the years 1999-2009, however, we see an increase to 8% in 2010-2016. Also, the number of studies that remained unclear was at 1% for 2017-2021 which is equivalent to that of 2010-2016 with none being unclear in 1999-2009.



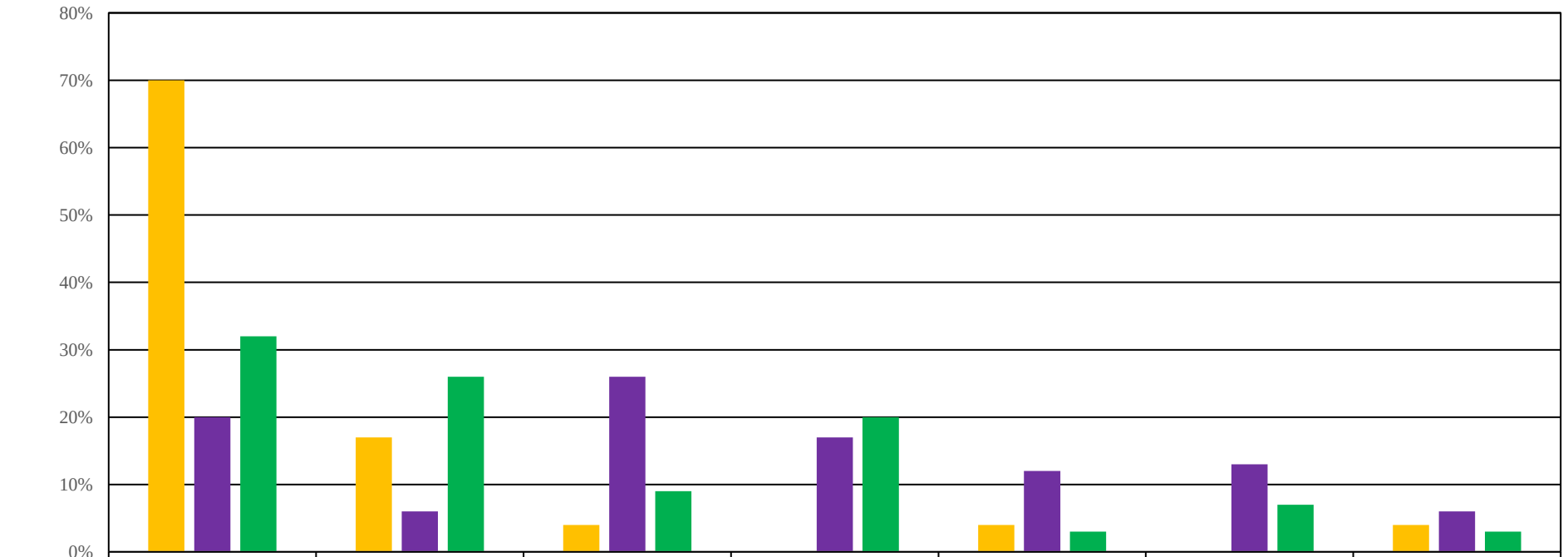
	Descriptive	Comparison	Grounded Theory	Statistical	Measurement	Tool Development	Other	Unclear
1999-2009	83%	8%	3%	2%	2%	2%	0%	0%
2010-2016	52%	17%	10%	8%	8%	1%	3%	1%
2017-2021	60%	5%	5%	10%	2%	5%	11%	1%

Figure 4.5 Distribution of means of analysis for 1999-2021, data up to 2016 was added from Vallon et al. (2018).

4.2.5 Distribution of study contribution

This section compares the distribution of study contributions for the full period 1999-2021. As shown in Figure 4.6 it is evident that contributions in the form of lessons learned are overall the most frequently seen contribution for the years 1999-2021. However, this study contribution had decreased significantly from 70% in 1999-2009 to 20% in 2010-2016 in favour of more rigorous contributions such as case study analysis (26%) and frameworks/models (17%). The new data shows an increase of contributions in the form of lessons learned to 32%. Although this is significantly lower compared to 1999-2009, it is 12 percentage points greater compared to 2010-2016. This may be due to the decrease in case study analysis from 26% in 2010-2016 to 9% in 2017-2021.

An important observation is that contributions in the form of recommendations/best practices have increased to 26% in 2017-2021 from 6% in 2010-2016 and 17% in 1999-2010. Furthermore, contributions in the form of frameworks or models are on the rise from no contribution in 1999-2009 to 17% in 2010-2016 and then 20% in 2017-2021. This 3% increase is minimal; however, it may indicate that contributions in the form of frameworks or models will be seen more in future studies. In addition, the new data reveals a decrease in contributions in the form of comparison or transformation, literature reviews, and tool development in 2017-2021. These aforementioned contributions are seldomly seen in studies.



	Lessons Learned	Reccomendations/Best Practices	Case Study Analysis	Framework or Model	Comparison or Transformation	Literature Review	Tool Development
1999-2009	70%	17%	4%	0%	4%	0%	4%
2010-2016	20%	6%	26%	17%	12%	13%	6%
2017-2021	32%	26%	9%	20%	3%	7%	3%

Figure 4.6 Distribution of study contributions for 1999-2021, data up to 2016 was added from Vallon et al. (2018).

4.3 Quantitative Analysis

In this section, the 98 empirical cases (out of the final set of 148 included studies) with successful applications and characteristics of agile practices in GSD are analyzed.

4.3.1 Empirical Characteristics

This section discusses the empirical characteristics of each of the successful cases of the recent period and compares them to the earlier periods. Table 4.1 shows the empirical characteristics of the successful cases for the years 1999-2021. It is evident that the majority of reported cases continue to focus on the SE process as a whole with 71% of cases in 2017-2021 compared to 64% in 2010-2016. This indicates that current research in agile practices in GSD remains relatively holistic and focuses less on specific areas as mentioned in Vallon et al. (2018). In the two earlier periods, certain empirical characteristics such as project size, project duration, and application duration were not frequently reported. However, in 2017-2021 the largest proportion of cases were small (42% of cases) in nature compared to 30% in 2010-2016 and 24% in 1999-2009. This indicates that research in GSD tends to favour projects that involve fewer participants. Moreover, these projects consisted of mainly industry practitioners with the exception of students in some cases. However, some contextual details such as project duration were not regularly defined (42% of cases were unclear in 2017-2021) and this trend is evident for the years 2010-2017 with 38% and 66% from 1999-2009.

Additionally, in all three of the periods, in a majority, development was globally distributed (as opposed to distributed within a single region or country). This implies that research in GSD is steadily growing because the number of cases, whereby, members were globally distributed had increased. Another important observation is the application domain in cases, and it is evident that many cases focused on enterprise software (37% in 2017-2021) compared to 9% in 2010-2016 and none in 1999-2009. The application domain remains largely unclear with 64% in 1999-2009 and 31% in 2010-2016 compared to 24% in 2017-2021.

Table 4.1: Empirical characteristics of successful cases for the years 1999-2021.

Period	Project Size	% of cases	Project Duration	% of cases	Global software development	% of cases	Knowledge Area	% of cases	Application Domain	% of cases
2017 - 2021	Large	22	Long	22	Yes	99	SE Process	71	Web	6
	Medium	20	Medium	26	No	1	SE Management	10	Enterprise Software	37
	Small	46	Short	9	Unclear	0	Tools & Methods	14	Telecommunication	4
	Unclear	10	Unclear	42			Testing	2	Service Provider	1
							Requirements	2	Finance	5
							Design	0	Open Source	1
							Construction	0	Automation	2
							Maintenance	0	Other	18
									Unclear	24
2019 - 2016	Large	25	Long	39	Yes	72	SE Process	64	Web	17
	Medium	22	Medium	20	No	16	SE Management	12	Enterprise Software	9
	Small	30	Short	2	Unclear	12	Tools & Methods	12	Telecommunication	8
	Unclear	22	Unclear	38			Testing	4	Service Provider	8
							Requirements	2	Finance	7
							Design	2	Open Source	4
							Construction	2	Automation	2
							Maintenance	0	Other	13
									Unclear	31
1999 - 2009	Large	15	Long	21	Yes	70	SE Process	15	Web	15
	Medium	17	Medium	11	No	2	SE Management	21	Enterprise Software	0
	Small	24	Short	0	Unclear	25	Tools & Methods	12	Telecommunication	6
	Unclear	44	Unclear	66			Testing	9	Service Provider	2
							Requirements	6	Finance	2
							Design	6	Open Source	0
							Construction	18	Automation	0
							Maintenance	12	Other	23
									Unclear	64

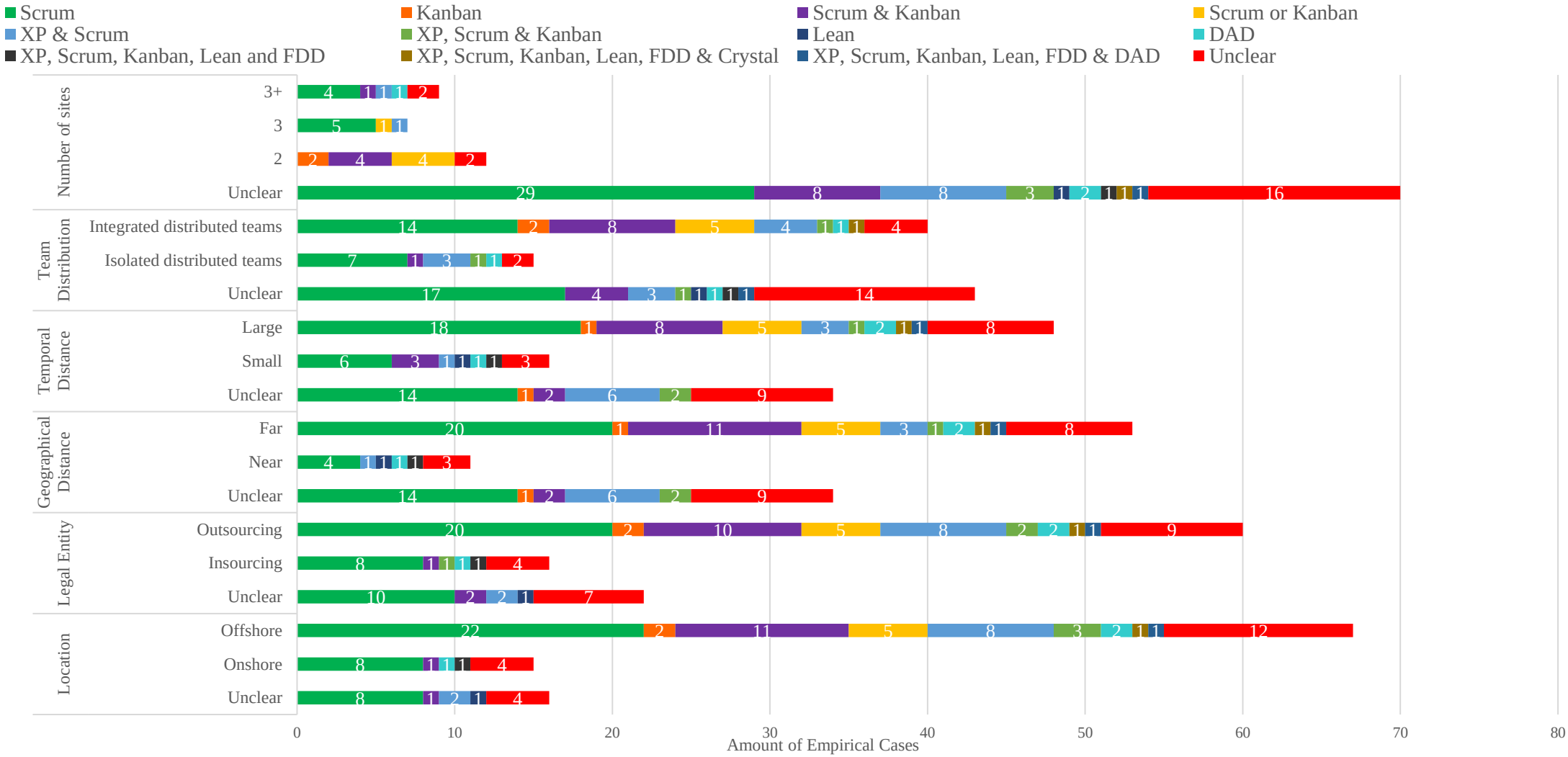


Figure 4.7 Usage of agile methods in various distirbution settings for the years 2017-2021.

4.3.2 Agile methods in various distribution settings

This section considers the various agile methods used in different distribution settings for the years 2017-2021 and compares these methods with those adopted in the earlier periods. In Figure 4.7 every part of the distribution category (location, legal entity, geographical distance, temporal distance, team distribution, and the number of sites) adds up to the total number of 98 successful empirical cases. For example, in the distribution setting *location*, 67 cases featured an offshore environment, 15 were onshore and the remaining 16 cases were classified as unclear as to whether they were offshore or onshore environments.

Considering the methodology adopted, Vallon et al. (2018) reported that Kanban or Lean software development was only applied in a small number of cases in GSD as well as FDD and DAD (Disciplined Agile Delivery). In 2017-2021, the overall picture is similar to Vallon et al. (2018) but different in some respects. For example, in Figure 4.10 it can be noticed that Scrum and Kanban have been used together (Scrumban) which was common for studies published between 2017-2021. This trend was not evident in Vallon et al. (2018). However, the new data support Vallon et al. (2018) in that Lean, DAD, FDD, and Crystal continue to remain a small portion of the cases in GSD. In 2017-2021 the agile methods employed in GSD are ranked as follows: Scrum (38 cases), Unclear agile method descriptions (20 cases), Scrum and Kanban (13 cases), XP and Scrum (10 cases), Scrum or Kanban (5 cases), XP, Scrum and Kanban (3 cases), DAD (3 cases), Kanban (2 cases), Lean (1 case), and combination of Scrum, XP, Kanban, Lean, FDD, DAD and crystal (1 each). Therefore, Scrum overall was the most popular agile method adopted followed by Scrum and Kanban practices used together. The following distribution details were most frequently adopted: offshore (67 cases), far distance (53 cases), two-site environment (12 cases), large temporal distance (48 cases), and outsourcing (60 cases). Not all cases reported details for each distribution category

According to Vallon et al. (2018)'s data, studies did not solely follow the XP method, however, a combination of XP and Scrum was adopted in multiple cases. This was somewhat different from Jalali and Wohlin (2012) where their findings pointed to the usage of XP independently. The new data from the most recent period, as depicted in Figure 4.9, supports Vallon et al. (2018) in that no studies used the XP method in isolation.

Vallon et al. (2018) discussed the distribution of agile methods for their study period (2010-2016), however, there was no comparison of agile methods for the earlier period (1999-2009). Therefore, Figures 4.8, 4.9, and 4.10 compare the distribution of agile methods for the years 1999-2021. Figure 4.8, which depicts the data for the period 1999-2008, shows that Scrum, XP and other agile methods (33% each) were equivalent in distribution. Therefore, according to Jalali and Wohlin (2012)'s data there were no agile methods more popular than the other. Also, XP method was used in isolation which was not evident for the later periods. According to Vallon et al. (2018)'s data, Figure 4.9 depicts that Scrum (60% of cases) was the most widely adopted agile method for the years 2010-2016, followed by XP and Scrum (16% of cases) and then unclear studies (13% of cases). Figure 4.10 depicts the distribution of agile methods for the years 2017-2021, it is evident that Scrum continues to be the dominant agile method (38% of cases). However, this is significantly less compared to the years 2010-2016. This decreased use of Scrum methods can be explained by the increased use of Scrum and Kanban (Scrumban) practices together. Therefore, the new data compared to the existing data reveals that recent studies are adopting more mixed agile approaches and more diversity since the original period within GSD.

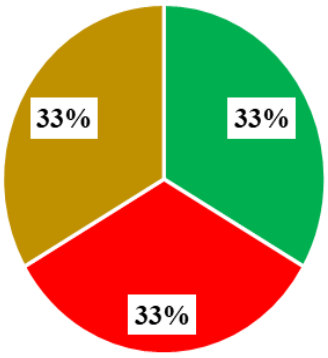


Figure 4.8 Distribution of agile methods for the years 1999-2009

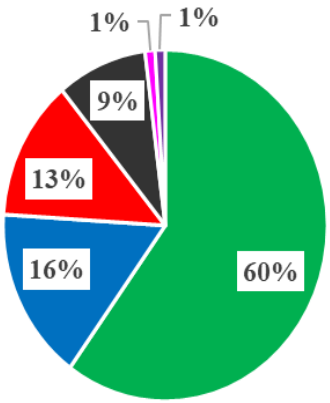


Figure 4.9 Distribution of agile methods for the years 2010-2016

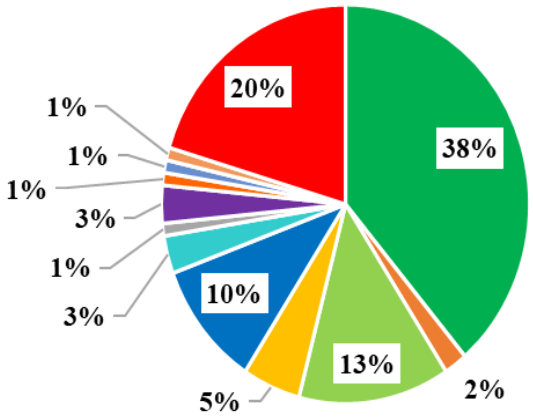


Figure 4.10 Distribution of agile methods for the years 2017-2021

- Scrum
- Kanban
- Scrum & Kanban
- Scrum or Kanban
- XP
- XP & Scrum
- XP, Scrum & Kanban
- Kanban/lean
- Lean
- FDD
- DAD
- XP, Scrum, Kanban, Lean and FDD
- XP, Scrum, Kanban, Lean, FDD & Crystal
- XP, Scrum, Kanban, Lean, FDD & DAD
- Unclear

4.3.3 Frequencies of agile practices

This section provides an overview of agile practices that have been successfully applied for the years 1999-2021 (see Appendix D for a full overview of agile practices compared across the years 1999-2021). Figures 4.11, 4.12, and 4.13 show the distribution of successfully applied agile practices in GSD for the years 1999-2021. It is important to note the term "agile practice" in this context refers to any element of an agile method such as Scrum and XP that performs some activity and was mentioned in a paper or report. For example, standup meetings and backlogs qualify as practices because they are a defined way of working with requirements (Vallon et al., 2018). Similarly to Vallon et al. (2018), the majority of studies did not report on success or failure for individual agile practices, therefore, by default, all agile practices were considered if applied in the context of an overall successful case.

From 1999-2021, the standup meeting has been the most frequently used agile practice with a total of (111.5 cases)[11]. Therefore, based on the analysis, standup meetings are a key element within the GSD domain, followed by sprint/iterations (100.5 cases) and backlog (87 cases). It is important to note that these agile practices (i.e standup meetings) in a distributed enviroment used technology to facilate communication between team members. The most commonly used ICTs in cases were Slack, Zoom, and Skype. Furthermore, the years 2010-2016 follow the same ranking with standup meeting (52 cases) then backlog (45 cases), and sprint/iterations (38 cases). The period 1999-2009 also featured standup meetings (18.5 cases) as the most frequently used agile practice but then followed by sprint/iterations (13.5 cases) and continuous integration (12 cases). However, the years 2017-2021 follow a different ranking with sprint/iterations (49 cases), followed by standup meetings (41 cases), and backlog (35 cases) as the second and

[11]Jalali and Wohlin (2012) mentioned half cases in their study, however, no particular explanation was provided for this phenomenon.

third most frequently applied agile practices. This indicates that sprint/iterations are becoming increasingly adopted and were considered an essential practice in GSD.

In comparison to the years 1999-2009, where six practices out of the top ten were Scrum-based (e.g. standup meetings, sprint/iterations, sprint planning etc), in 2010-2016 eight of the top ten practices are affiliated with Scrum, and XP practices (e.g. standup meetings, backlog, sprint/iterations etc) have been used less compared to Scrum practices. This indicates that the majority of studies in 2010-2016 used mixed approaches with Scrum. This observation is also supported for the years 2017-2021, where Figure 4.10 shows the use of Scrum and Kanban in studies. However, it is evident that Scrum-based agile practices remain dominant for the years 2017-2021.

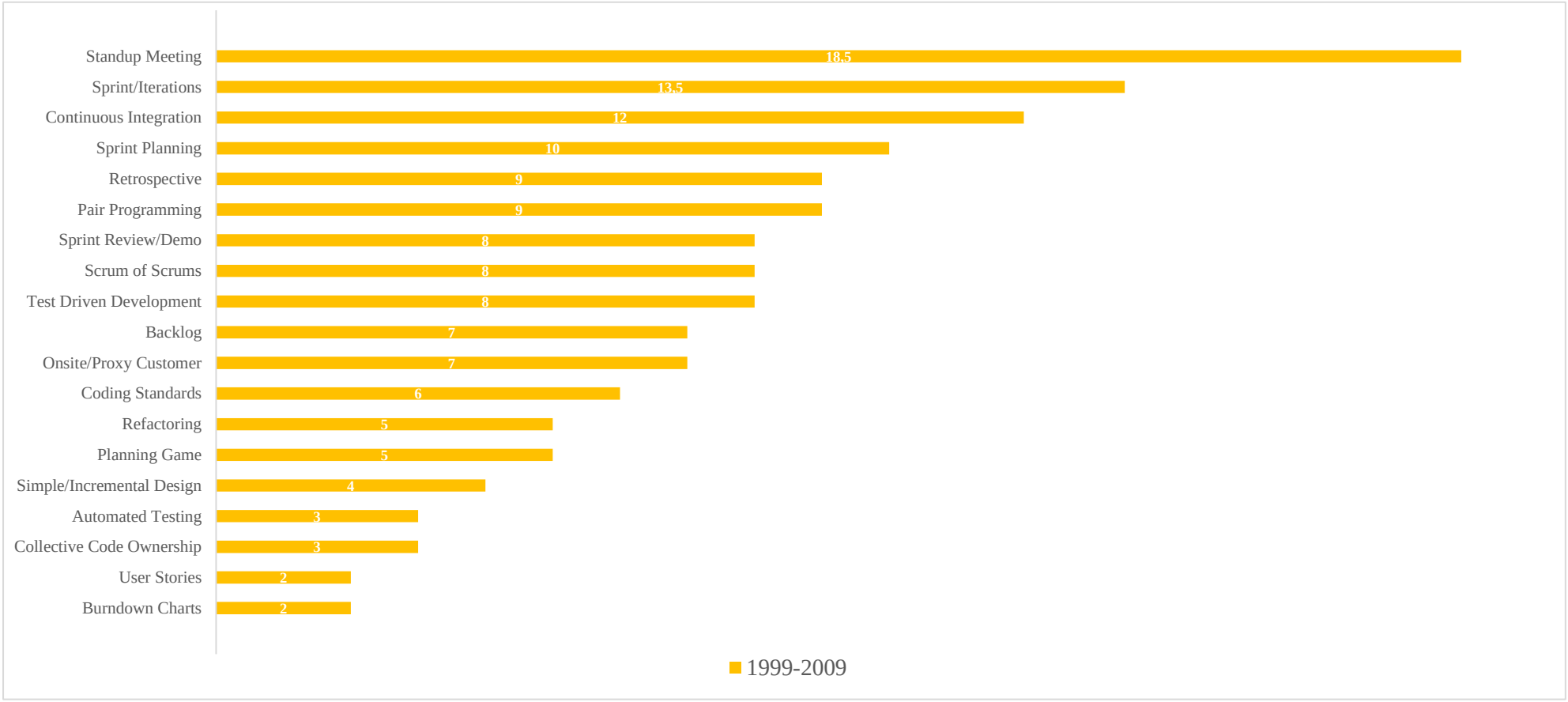


Figure 4.11 Overview of agile practices that have successfully been applied for the years 1999-2009, data up to 2009 was added from Jalali and Wohlin (2012).

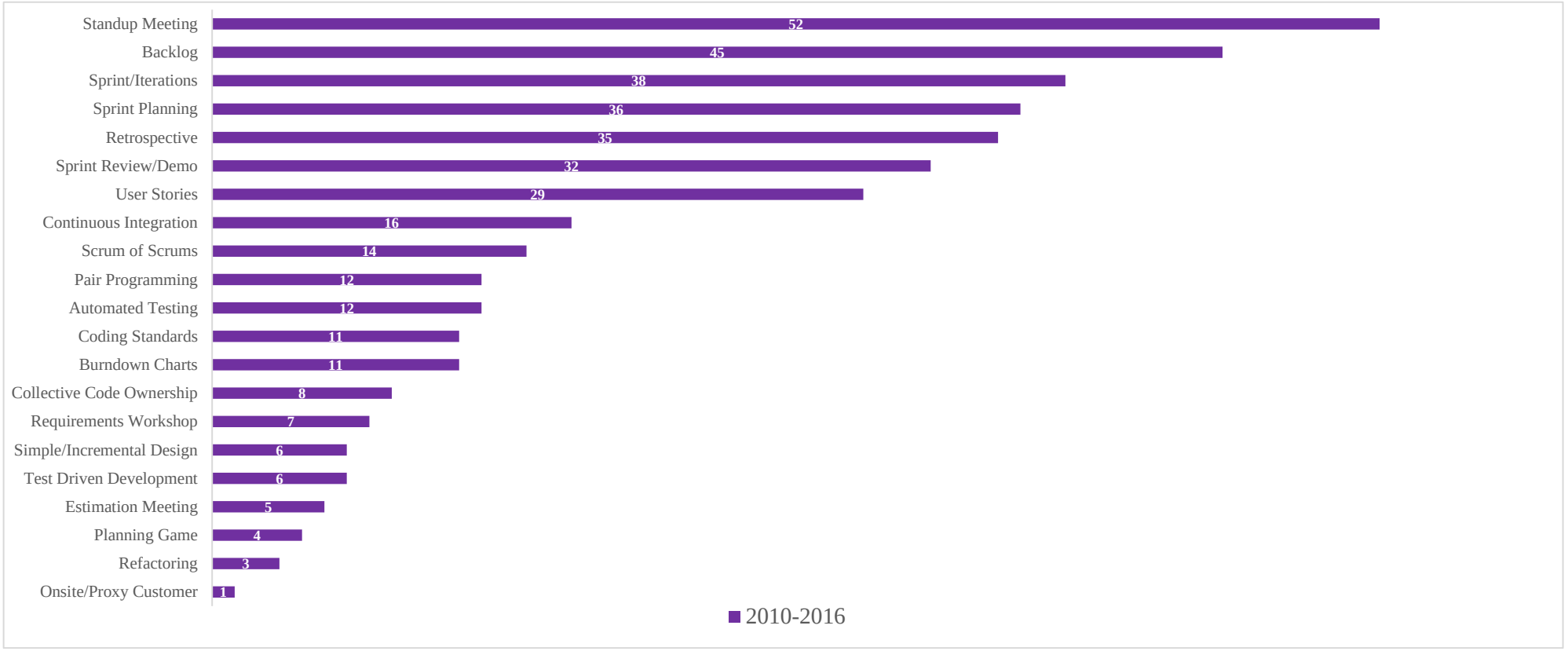


Figure 4.12 Overview of agile practices that have successfully been applied for the years 2010-2016, data up to 2016 was added from Vallon et al. (2018).

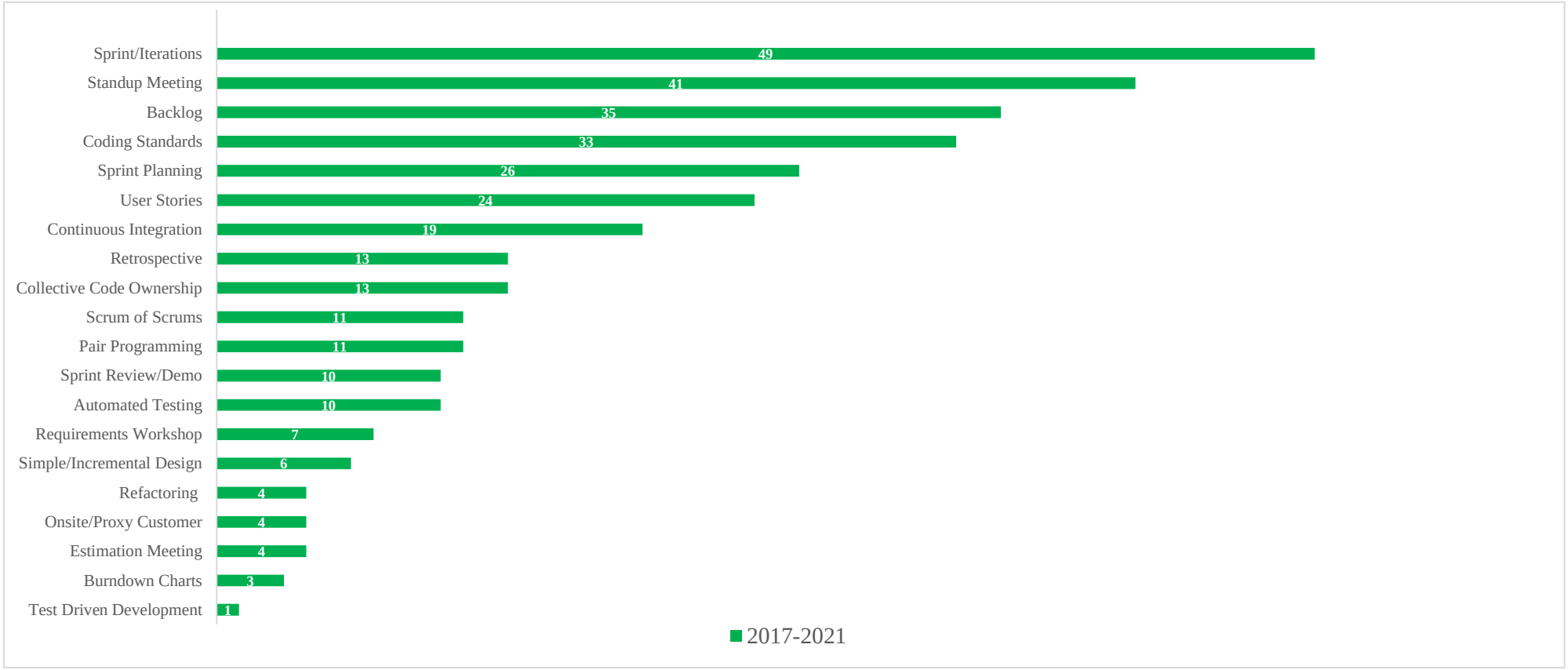


Figure 4.13 Overview of agile practices that have successfully been applied for the years 2017-2021.

4.4 Qualitative Analysis

This section addresses a holistic view of 1999-2021 (RQ1a and RQ2a) as a continuation to Jalali and Wohlin (2012) and Vallon et al. (2018) and compares the new studied period 2017-2021 to the earlier period results of 1999-2016 (RQ1a and RQ1b). Furthermore, this section will uncover trends for the full period 1999-2021 by analyzing and comparing the trends identified by Jalali and Wohlin (2012), and Vallon et al. (2018) to determine whether those trends continue to be relevant or not. In addition, new trends are identified from the period 2017-2021.

4.4.1 RQ1a: *What is reported in peer-reviewed literature about agile practices in software development in remote working scenarios between 1999 and 2021?*

This section explores the continuing trends across the whole period of 1999-2021 and describes various high-level themes that characterize this domain.

4.4.1.1 Challenge Categorizations

Following Jalali and Wohlin (2012), and Vallon et al. (2018) the implementation of agile practices in GSD comes with a variety of challenges. This section discusses studies that identify challenges with the implementation of agile practices in GSD and proposed solutions for these challenges. Through the systematic search for the years 2017-2021 numerous papers had identified similar challenges and recommendations in GSD, these common challenges were the most prevalent across all the studies, Alsahli (2017), Vithana et al. (2018), Batool, (2019), Manjavacas et al. (2020).

Many studies reported on several GSD challenges and mentioned numerous agile practices to counter these challenges. The common challenges identified were mainly communication, coordination, control, and were a result of temporal, socio-cultural, and geographical distances (Alsahil, 2017). The main identified agile practices that can aid with these challenges were sprint reviews, daily standups, iterations, and backlog. In addition, according to Vithana et al. (2018) the following key challenges had a significant relationship with project success. Communication barriers are the most discussed in literature (i.e., Alsahil, 2017; Batool, 2019)

because efficient communication is essential for project work. This can be difficult to exercise within remote situations. Therefore, the use of ICTs can mitigate against these communication challenges. The next challenge concerned the team's technical competency, the team's ability to carry out work on its own. Followed by customer involvement throughout the project which is important as it has a positive impact on project success due to customer needs being met.

In addition, ensuring that each member of the team has the required knowledge to perform tasks is crucial. Hence, when members are geographically distributed, coordinating, and controlling work is essential for project success. Thus, technology is vital when carrying out work especially when team members are distributed (Henry et al., 2021). It appears that communication, control, and coordination are the most prevalent GSD challenges in the literature (Alsahil, 2017; Vithana et al., 2018). Similarly, Batool (2018)'s study identified key challenges of agile software development such as communication, language barriers, culture diversity, lack of documentation, time difference, geographical distance, temporal distance, socio-cultural distances, and knowledge management.

Communicating and working on projects where team members are globally distributed can be challenging due to different time zones and the lack of face-to-face interaction. Along with globally distributed teams comes the challenges of language barriers and culture diversity (Batool, 2018). Language barriers and culture diversity are particularly seen in cases where teams are distributed across many locations. Therefore, team members need to be accustomed to dealing with different types of people from different countries. These aforementioned challenges are similar to the challenges identified by Vallon et al. (2018). Additionally, Manjavacas (2020) highlighted some challenges concerning GSD governance, for example, agile principles, information, culture, ethics, behaviour, people, skills, and competencies. This implies that challenges of distributed teams are multi-folded and do not only face geographical and communication challenges but have to deal with different cultures, and individual beliefs as well as the level of competency of each team member.

To summarize these studies, a majority of the identified challenges were similar across all studies and had similar challenge categorizations to that of Vallon et al. (2018). The most common challenge across studies was communication where team members are distributed across several locations with different time zones. Therefore, the challenge exists in developing modified agile practices that are used for distributed situations and ensure that the use of technology/tools to communicate with members that are distributed is efficient and effective. The use of ICTs such as Slack, Skype or Zoom and other communication tools are used to mediate communication within distributed environments where time, geographical and temporal differences present a challenge (Alsahil, 2017; Batool, 2018).

4.4.1.2 Success Reports

The majority of studies published between 1999-2016 discussed successful cases. Similarly, in the more recent period, only a very small proportion of studies were unsuccessful cases. It is unclear if this is due to various publication biases, or whether it is a true reflection of reality. However, while the reason is unclear, important lessons can be learned by briefly considering those that have been reported. Rajpal (2018)'s case report described various shortcomings and the team was unfamiliar with agile practices (i.e. pair programming) within GSD. Therefore, the lack of experience and uncertainty led to an unsuccessful project. Paasivaara and Lassenius (2016)'s empirical study accounts for a failed collaboration in the GSD project and found that tools and collaborative practices used in coordinating and controlling work in the project to be ineffective, contributing to the failure of the agile implementation. In addition, Paasivaara and Lassenius (2016)'s study adopted agile development practices (i.e., sprints, backlog, retrospectives) and found these practices incapable of dealing with collaborative issues in globally distributed work because the mindset of agile was partially missing. Further, the product was difficult to divide into various requirement areas and there was a lack of knowledge in Scrum implementation which resulted in unsuccessful application of agile practices.

To contrast with these failed cases, while the comparative analysis provides a high-level picture, valuable lessons can also be developed by focusing briefly on two-key successful cases. Guillot et al. (2017)'s multi-case study showed detailed descriptions of how Scrum practices were used in three cases and were successful despite of the challenges of GSD. The agile/Scrum practices that aided in the success of the case studies had early and frequent delivery of working software, customer-centric principle for meeting demands, and the project team were able to respond to change rapidly. Effective communication, organization and motivated members also played a crucial role for project success. Lous et al. (2018) describe a case study of a software development team that successfully adopted agile practices (i.e., standup meetings, sprint planning, etc.) while being distributed across several locations. Lous et al. (2018)'s case study was successful in that the project team designed and carefully managed a virtual work environment in which tools and agile practices were modified to aid the distributed development team.

In addition, Lous et al. (2018)'s case report was particularly successful because the project team was able to overcome common GSD challenges such as lack of attendance (this challenge was addressed by Daily-standup meetings and participation via online platforms). Lack of team cohesion was addressed with improved cooperation facilitated via Slack, and project manager calls to mitigate these issues. In addition, all team members had access to documentation through Github. This created an environment where members felt inclusive and valued. More importantly, team members shared responsibility and discussed any project challenges during retrospective meetings. Furthermore, knowledge sharing was another contributing factor to project success which was achieved through established practices and dedicated Slack channels. These channels of communication were designed to maximize transparency for members with different cultural backgrounds and mother tongues within the team, despite temporal and physical distances.

4.4.1.3 Lack of Contextual Information

Based on the data extraction process, the continuation of the trend reported by Jalali and Wohlin (2012) and Vallon et al. (2018) holds in that contextual information was not always explicitly stated in studies for the years 2017-2021. The common categories of contextual information that were rarely defined in studies were project duration, geographical distance, temporal distance, number of sites/locations, and whether the teams were integrated or isolated. To address the lack of contextual information among studies, authors should represent this information in the form of a table. The data extraction checklist (see Appendix B) drives the description of contextual information and other researchers should make use of such checklist to report a common set of elements present in all studies. This will improve comparability and help mature the field.

4.4.1.4 Increasing Popularity

According to Vallon et al. (2018)'s data, there is a growing interest in agile GSD and it is becoming relatively popular over the years with an average of twenty publications per year from 2009-2016. The new data confirms this trend, and it was found that an average of 30 studies were published per year from 2017-2021. This growing trend will bring along a variety of empirical characteristics (Table 4.1) and distribution scenarios (Figure 4.7) and successfully applied agile practices in GSD (Figures 4.11, 4.12, and 4.13). As the number of publications in agile GSD increases the focus on future research will contribute to new findings, and elements within this domain. This finding is also supported by the results from the 15th annual state of the agile report, where agile adoption within software development teams is increasing from 37% in 2020 to 86% in 2021. Furthermore, the survey indicated that due to the Covid-19 pandemic, 25% of respondents indicated that remote work would remain while the remaining proportion of respondents believed in a hybrid approach with a partial return to office work.[12]

[12]https://digital.ai/resource-center/analyst-reports/state-of-agile-report

4.4.1.5 Modified Agile Practices

Traditionally, agile practices have been used in co-located teams where face-to-face interaction between team members exists. However, with remote work on the rise in recent years, there is a need for modified agile practices to work in globally distributed environments (Vallon et al., 2018). This necessity has been identified by Jalali and Wohlin (2012) and Vallon et al. (2018). This need is also evident for the period 2017-2021. Stray and Moe (2020) explored the use of communication tools such as Slack and video conferencing where teams were distributed across several locations to coordinate agile practices such as daily standups, retrospective meetings, and sprint reviews. Furthermore, the lack of awareness when teams are distributed makes daily feedback challenging. Therefore, the use of asynchronous communication is important.

In addition, Lous et al. (2018) referred to various communication media used to facilitate communication between team members in distributed environments. Slack's screen sharing feature and Github aid in the remote implementation of technical agile practices such as pair programming and code reviews. Slack calls are also useful for organizational agile practices such as standup meetings, retrospectives, task allocation and one-on-one meetings (Lous et al. 2018). Other tools like Google Products (i.e. sheets) are useful for communication, and collaboration purposes.

The literature review by Camara et al. (2020) mentioned that communication practices in GSD intend to reduce misunderstanding and enhance the relationships among distributed team members through sharing common information across different locations. The study also highlighted some of the key asynchronous and synchronous communication tools (i.e. emails, telephone, video conferencing, audio conferencing and Skype) that were used to conduct, manage, and monitor estimation meetings, planning sessions, retrospectives, sprints, sprint reviews, backlogs, pair programming and code reviews when teams and Scrum masters were globally distributed (Camara et al., 2020). Furthermore, Scrum of Scrums was used to keep teams updated on the events of the project and other practices like Kanban

boards promoted visualization and allowed each distributed member to have their own physical Kanban board. Electronic boards were used for all distributed members to visualize the status of updates of tasks (Camara et al., 2020).

4.4.1.6 Dominance of Scrum

According to Vallon et al. (2018)'s data the most frequently used agile method is Scrum followed by XP and Scrum for the years 2010-2016. XP practices were featured across several studies (9 cases from 1999-2009) and several more for 2010-2016. However, for the period 2017-2021, this trend is somewhat relevant, but some new changes have manifested. The new data represented in Figure 4.10 show that Scrum was used in 38% of cases from 2017-2021 (e.g. Srivastava & Jain, 2017; Chilito et al., 2018; Pardo Calvache et al., 2019) and continues to be the dominant agile method. Therefore, with Scrum practices being relatively researched and matured, Scrum becomes a promising agile method for organizations to adopt particularly in remote situations (Srivastava & Jain, 2017).

The combination of Scrum and Kanban (Scrumban) practices (13% of cases) from 2017-2021 (e.g. Banijamali et al., 2017; Godoy et al., 2019; Beecham et al., 2021) has grown and exceeded XP and Scrum practices. The advantages of Scrumban such as iterative development, and limited work-in-progress, would alleviate the challenges of synchronization between distributed sites, communication, and culture (Banijamali et al., 2017). Due to the number of positives when using Scrumban in remote situations it may lead to more projects adopting mixed-method approaches. This trend is also supported by Godoy et al. (2019) who mentioned that the use of Scrum and Kanban practices allowed for team task allocations and effective communication between distributed members. Therefore, Scrumban practices are well suited for projects under remote conditions. To further support this trend, in recent times, it was found that Scrum and the combination of Scrum, and Kanban practices were the dominant agile methods. This is an indication that mixed-method approaches are a growing interest in the GSD domain and there is a shift from traditional practices to more mixed-method approaches to deal with challenges of globally distributed work.

4.4.1.7 Experience Reports and Evaluation Studies

Experience reports were the most frequent research type for the years 1999-2009 and were also published for the period 2010-2016. However, the number of studies that were experience reports was significantly less for later periods compared to the earlier periods. According to Vallon et al. (2018), some experience reports varied in quality with some providing a high-level overview of how agile practices were implemented in GSD. It is important to note that, given the nature of experience reports and their anecdotal characteristic, such type of studies can be relatively unclear, and certain contextual information may not be immediately obvious. In Figure 4.2 it is evident that the minority of studies were experience reports (14 cases) from 2017-2021 (i.e., Moe et al., 2017; Rajpal, 2018). The study by Moe et al. (2017) presented experience from developing and maintaining team knowledge for global virtual teams that were distributed between Poland and Norway. This verifies that the number of studies that are experience reports are decreasing due to the lack of covered related work and repetitions as mentioned by Jalali and Wohlin (2012).

The new data reveals that fewer experience reports were published in 2017-2021, therefore, the problem of lack of contextual information in experience reports becomes less significant. However, the most frequent research type for the period 2010-2016 were evaluation studies and this trend continues for the years 2017-2021 (i.e., Bick et al., 2018; Hossain et al., 2019; Smite et al., 2020). Bick et al. (2018)'s study used case study analysis as part of their research design and studied a large software development team that were distributed over four locations, due to the nature of methodology used it is considered an evaluation study. Similarly, Hossain et al. (2019) was an evaluation study because the data was collected by interviews, these interviews were conducted using communication media (i.e., Skype) as participants were distributed across several locations. In the end, studies at large used data collection techniques that involved surveys, interviews, case studies and questionnaires. Hence, the maturing field is moving away from experience reports.

4.4.1.8 Case studies exceeded models/frameworks

According to Jalali and Wohlin (2012) and Vallon et al. (2018)'s data many case studies were published from 1999-2016 (Figure 4.4) and various literature reviews in the period 2010-2016. In Figure 4.4 it is evident that case studies from 1999-2016 were more dominant than literature reviews. Also, from 2010-2016 most studies were case study based and lessons learned rather than the development of frameworks and models. However, as for 2017-2021, this trend does not hold. The opposite was identified where the number of studies (i.e., Noll et al., 2017; Godoy et al., 2019; Cruz et al., 2020) focusing on the development of frameworks/models such as *Scaled Agile frameworks, Agile Blueprint models* had exceeded the number of studies that analysed individual case studies (i.e., dos Santos et al., 2018; Szabo & Steghofer, 2019). Again, this is a sign of a maturing research field. In Figure 4.6, it is evident from 2017-2021 that 20% of study contributions were frameworks/models compared to 9% that were case study analysis. Therefore, the trend of case studies over frameworks/models falls short. Furthermore, in Figure 4.4 literature reviews (22% of cases) were slightly more prominent compared to case studies (18% of cases) for the years 2017-2021. Hence, it can be observed that there is a subtle shift from case studies to more literature reviews in the later periods. The majority of the literature reviews (i.e., Hoda et al., 2017; Akbar et al., 2019; Noor et al., 2021) from 2017-2021 contributed in terms of lessons learned and recommendations.

4.4.1.9 Lack of details on success/failure

It was found that 98 cases reported overall success and only 6 failures in 2017-2021. Jalali and Wohlin (2012) mentioned that the number of cases that were failures made a significantly small proportion of cases in 1999-2009. According to Vallon et al. (2018)'s study, the trend was similar in that 89 cases reported overall success and only 4 failures. Therefore, the common theme that exists for the full period (1999-2021) is that failure cases make up a relatively small percentage of studies with the majority being reported as successful. However, it is not clear if this is due to publication bias or a reflection of the actual proportion of cases that are failures.

Many studies focused on challenges and solutions in general in GSD, however, in case study research very few explicitly state whether the cases were a success or failure. Therefore, during the data extraction process, it was difficult to identify and extract from the full text whether the study was a success as no criteria or information indicated the details on success/failure. To this end, it is suggested that authors define these criteria upfront in future case studies on agile GSD. Ultimately, this will improve documentation of the success or failure of individual agile practices in GSD and improve the quality of research output for future studies within the GSD research area.

4.4.2 RQ1b: *Compared to the period 1999 - 2016, what reporting differences exist in the peer-reviewed literature about agile practices in remote working scenarios for the period 2017-2021?*

This section explores the comparison aspect of this study, it reflects on how the studied period 2017-2021 differs from 1999-2016 studied by Jalali and Wohlin (2012) and Vallon et al. (2018).

4.4.2.1 Scaled up agile

According to Jalali and Wohlin (2012) there was insufficient evidence to support that agile is efficiently applicable to large-scale environments for the years 1999-2009. However, the years 2010-2016 showed evidence of successful large-scale applications in GSD, as Table 4.1 showed that 25% of cases were reported with large project sizes, considerably more than 15% of cases from 1999-2009. The years 2017-2021 showed that 22% of cases were reported with large project sizes. This is slightly less compared to the years 2010-2016, however, it is not a significant decrease. Therefore, the data reveals in the later periods that agile was efficiently applied in large-scale environments. Examples for large-scale agile GSD adoptions for the years 2017-2021 include studies such as Bick et al. (2018) and Marek et al. (2021). For instance, Bick et al. (2018)’s study reports a case in which hybrid approaches were adopted in a large-scale development project that developed complex and successful standard enterprise software across four locations spread

over Germany, India, and China. Similarly, Marek et al. (2021) used a survey-based research design for participants within international software development companies that were distributed across several countries worldwide. The most common agile method adopted was Scrum and scaled Scrum frameworks (i.e., SAFe).

4.4.2.2 Teaching activity

Jalali and Wohlin (2012) did not report on teaching activities in this context between 1999-2009. However, Vallon et al. (2018) found teaching activities in studies for the years 2010-2016. Published teaching studies included lessons learned and course results to teach the way of agile practices in GSD. Teaching activities were also evident for the years 2017-2021 to expose students to the application of agile practices in distributed environments. Notable examples for the period 2017-2021 include Defoulas et al. (2017) with software engineering student teams distributed across several countries (i.e., Egypt, Pakistan, Palestine) that involved a simulation of virtual GSE teams working together to perform various software development tasks (e.g., Scrum meetings). The aim was to understand GSE team member collaboration. Rzhvesky et al. (2020) proposed a distributed team that requested students to work in a distributed team across two distinct locations (i.e., Portugal and Slovenia) with the aim to diagnose communication behaviour in remote working teams. The main distinction between industry studies and teaching-based studies is that teaching activity is based on experience of knowledge, whereas the objective of industry studies is to work towards answering specific research questions and contributing to a specific area of research. While on the contrary, teaching studies aim to educate and inform students with limited or no experience within a specific research field.

4.4.2.3 GSD Terminology

According to Jalali and Wohlin (2012), the terminology used was relatively diverse in nature in their analysed studies and practitioners and researchers needed to reach some agreement of common terminology in GSD. From 1999-2009, the majority

of cases reported on "distributed teams", and "offshore" and "agile" were used without further contextual details. Similarly, in Vallon et al. (2018) the empirical characteristics were not fully described, however, large improvement had been made compared to studies published from 1999-2009. Later periods allowed for the analysis of various distribution scenarios as illustrated in Figure 4.7. The period 2017-2021 revealed that many studies use common GSD terminology, however, in some cases "distributed teams" would be replaced by "remote teams" or "onshore" would be "localized". Therefore, not all studies used similar GSD terminology, but within the GSD context, these terms have identical meanings. Future studies need to standardize the terminology used within the GSD research field to avoid ambiguity. Similarly, to Vallon et al. (2018) the data extraction checklist (Appendix B) incorporates GSD taxonomy.

4.4.2.4 Continued decline in XP practices

Jalali and Wohlin (2012) found that XP was the main featured agile method followed by Scrum for the years 1999-2009. However, Figure 4.9 shows that the overall pattern shifts from 2010-2016 where not a single case used XP in isolation in GSD. Vallon et al. (2018) found that 16% of cases used Scrum as their agile method but were complemented by XP development practices. Therefore, XP was no longer the center of agile implementations in GSD. In the period 2017-2021, the overall picture is quite different compared to the years 1999-2016 where a decline in the adoption of XP practices is noticeable. In Figure 4.10 it is evident that Scrum and Kanban practices had increased in popularity and were one of the main contributing agile method combinations in GSD from 2017-2021. Scrum and XP practices only made 10% of cases which was significantly lower compared to the earlier periods. Therefore, the adoption of XP practices in GSD from 2017-2021 was declining and being surpassed by Scrum and Kanban (Scrumban) practices (i.e., Banijamali et al., 2017; Godoy et al., 2019), respectively. According to Banijamali et al. (2017) Scrumban combines both Scrum and Kanban to create a management framework for improving software engineering practices. Banijamali et al. (2017)'s study revealed that the combination of Scrum and Kanban to facilitate

coordination among distributed sites was relatively unknown because limited research has been performed on the capability of Scrumban within GSD during earlier periods, however, this is slowly changing due to the growing of research within the GSD domain. Similarly, Godoy et al. (2019) focused on developing and designing a new software development model called a blueprint, a model that combines Scrum and Kanban methodologies to facilitate team task allocations and effective communication between members in GSD.

4.4.2.5 Agile GSD during the Covid19 Pandemic

The new data reveals Covid related challenges/changes within GSD. This would have not been reported by Jalali and Wohlin (2012) and Vallon et al. (2018). The Covid-19 pandemic had a fundamental impact on agile practices and many changes had to happen to deal with the pandemic. Only one study included in the present review reported on changes during the pandemic. Merek et al. (2021)'s study emphasizes that agile software development teams had to rapidly transition to remote work and adapt to new business circumstances. However, according to Marek et al. (2021)'s survey only some agile software developments were able to make a smooth transition due to working experience in distributed environments. In full remote situations, the use of different methodologies (i.e., Scaled Scrum) and tools (i.e., Slack) enabled the teams to communicate with each other across several locations.

4.4.2.6 Transition and Transformation

Jalali and Wohlin (2021) did not report on transition and transformation studies (guided by learning and development) in 1999-2009. In 2010-2016, according to Vallon et al. (2018), multiple studies covered the transition to agile GSD environments. For the period 2017-2021, there were not many cases that reported on transition and transformation (i.e., Roman et al., 2017; Roopa et al., 2018; Merek et al., 2021). Roman et al. (2017)'s core aim was to transition away from traditional methods in developing software to adopting agile philosophies and methodologies. Traditional methods proved incapable of dealing with the challenges of remote

work, while agile methods provided greater adaptability. Similarly, Roopa et al. (2018) shared the experience of a globally distributed software development team that transitioned from plan-driven approaches to lean methodologies and found the new practices to support an effective global team by improving trust, increasing communication, and encouraging decisions. According to Merek et al. (2021), transition and transformation was not a significant challenge because the globally distributed team was able to make the transition to remote work relatively easily due to their experience in working across several distributed locations.

4.4.2.7 Tool Support

In 2010-2016 with a maturing research field and improved knowledge on the successful implementation of agile practices in GSD, many tools were used along the way. In 2017-2021, many cases used some form of communication tools and technologies to perform work across several locations. Examples include Raith et al. (2017), Stray and Moe (2020) and Marek et al. (2021). Raith et al. (2017)'s data revealed that the majority of the distributed software development team's activities were supported by communication and collaboration tools (i.e., Microsoft Lync sessions, Microsoft OneNote), and file sharing tools (i.e., SkyDrive). Similarly, Stray and Moe (2020)'s case study showed that collaboration tools (i.e., Slack, Instant messaging) in distributed teams increased team awareness and can reduce the challenges of geographical distance allowing team members in global projects to communicate with each other. Moreover, Marek et al. (2021) found that various tools and frameworks helped the software development team transition to remote work during the Covid-19 pandemic in 2020. The most common collaboration tools used were Jira, Confluence, and the most used communication tools used were Teams, Slack and Github. These tools allowed for team members across different remote locations to collaborate and communicate.

4.4.2.8 Research types and contributions

Figure 4.2 depicts that in the later studied periods there was a transition from experience reports, which was the most frequent published research type from

1999-2009, to evaluation studies in 2010-2021. Examples of the latter include: Guillot et al. (2017), Arumugam et al. (2018) and Smite et al. (2020) indicating a maturing research field. According to Vallon et al. (2018) in 2010-2016 most studies used qualitative approaches in 52% of studies compared to 88% in 1999-2009. Furthermore, 22% of studies in 2010-2016 used mixed approaches up from 4% in 1999-2009 and 13% of cases used quantitative ones up from 2% in 1999-2009. In Figure 4.3 the number of studies that were qualitative was 40% in 2017-2021 this is significantly lower compared to the years 2010-2016. However, the number of studies that adopted mixed approaches was 33% in 2017-2021 up from 22% in 2010-2016. Therefore, the new studied period reveals that more cases are adopting mixed approaches in favour of qualitative approaches.

In addition, quantitative studies made up 9% of cases in 2017-2021 which is down from 13% in 2010-2016. As Figure 4.4 shows, there is a significant rise of studies being literature reviews from 1% in 1999-2009 to 15% from 2010-2016 followed by 22% in 2017-2021. Therefore, literature reviews are becoming increasingly popular within the GSD research domain which may imply a need for more studies that synthesize existing and current research in GSD. In Figure 4.5 the change in trends continues where less descriptive means of analysis was found with 52% in 2010-2016 and 60% in 2017-2021 compared to 83% in 1999-2009, however, more comparisons, grounded theory approaches, measurements, and statistical means were prominent in the later periods.

The distribution of study contributions for the years 1999-2021 is illustrated in Figure 4.6. where fewer lessons learned (20% of cases) in 2010-2016 and 32% of cases (i.e., Cruzes et al., 2017; Bick et al., 2018; Lautert et al., 2019) in 2017-2021 as their main contribution. This is significantly less compared to 70% from 1999-2009. For the years 2010-2016 case study analysis was the highest study contribution at 26% of cases compared to 9% in 2017-2021 and 4% from 1999-2009. The development of frameworks or models at 17% of cases from 2010-2016 up to 20% in 2017-2021 (i.e., Awar et al., 2017; Khan et al., 2019; Esteki et al., 2020). Furthermore, the new studied period reveals a trend where 26% of cases

contributed through recommendations/ practices this is significantly more compared to 6% in 2010-2016 and 17% from 1999-2009. This is indicative of a growing research domain where researchers can recommend best practices in GSD.

4.4.3 RQ2a: *Which agile practices, in which GSD settings, under which circumstances have been successfully applied in peer-reviewed research literature between 1999 and 2021?*

This section highlights which agile practices are used under different GSD settings for the years 1999-2021.

4.4.3.1 Countries Involved

There were numerous successful cases for the years 2017-2021 where teams were distributed across several countries. Key examples are the multiple case studies: Guillot et al. (2017); Smite et al. (2020); Beecham et al. (2021). Some offshore and onshore countries that were involved across these case studies were namely: USA, Canada, India, Australia, Denmark, UK, and Sweden. In 2017-2021 a majority of countries had large temporal and geographical distances between them. For example, in some cases team members were distributed among two sites, India and Sweden, and others over multiple sites, Australia, India and USA. Therefore, industry experts and practitioners came together around the globe to partake in software development projects and communicated through the use of tools. However, in some cases (i.e., Borrego et al., 2017; do Santos et al., 2018 & Khan et al., 2021) the focus was on single countries where participants/team members were insourced. Borrego et al. (2017) conducted an evaluation study wherein participants were from five Mexican software development companies across different cities within Mexico. Similarly, dos Santos et al. (2018) studied a Brazilian company that specialised in software development, therefore, in this case only single company in Brazil is evaluated. Khan et al. (2021)'s study collects data from practitioners working specifically in the Chinese GSD industry. It was noted that cases that focused on multiple locations provided an overall picture when team members faced temporal and geographical challenges. However, in cases that

focused on single countries, the aim was to better understand agile software development practices practitioners/members within close proximity.

4.4.3.2 Efficient agile methods, practices, and distribution type combinations

According to Figure 4.7, various patterns/combinations were prominent in successful cases. In the period 2010-2016, it was evident that Scrum, and the combination of XP, and Scrum methods were the most frequently adopted agile methods. The combination of XP and Scrum appeared in 16% of cases for the years 2010-2016. However, according to Figure 4.10 in 2017-2021 it was found that Scrum and XP practices had declined. This decline was explained by the increased usage of Scrum and Kanban (Scrumban) practices where 13% of cases used Scrum and Kanban practices. In some situations, both Scrum and Kanban practices were complimented by XP (3% of cases). Additionally, for the years 2017-2021, in the minority of studies (2% of cases) it was found that an amalgamation of different agile methods (i.e., XP, Scrum, Kanban, Lean, FDD, DAD and Crystal were used. These new combinations were not reported by Vallon et al. (2018) and Jalali and Wohlin (2012). Therefore, it was not possible to compare these combinations of agile methods to earlier periods. In the end, with Scrum being the largely dominant agile method for the years 1999-2021, the agile practices that appeared in most cases were Scrum-orientated practices (i.e., sprint/iterations, standup meetings, backlog, sprint planning). Important to note, most cases used a variety of agile practices together. For example, the most popular combination of agile practices were sprints, backlog, sprint planning, sprint reviews, retrospectives and daily stand-ups.

4.4.4 RQ2b: *How do the results differ for the period 2017-2021 compared to those of 1999-2016?*

This section analyses the differences for the period 2017-2021 in terms of countries involved and the types of methods and distribution scenarios compared to the years 1999-2016.

4.4.4.1 Differences in countries involved

The top countries involved in the supplier to customer relationship[13] did not change for the years 2010-2016. In the period 2017-2021, the majority of countries in the studies were similar to that of Vallon et al. (2018) where many outsourcing vendors were based in India, USA, and Canada. Therefore, not many differences in terms of countries involved between the two studied periods.

4.4.4.2 Differences in the types of methods/distribution scenarios

Jalali and Wohlin (2012) encountered studies with vague descriptions in their analysed cases from 1999-2009. For the period 2010-2016, not all the details about distribution scenarios were consistently reported. However, a comparison can be made highlighting the differences between Vallon et al. (2018)'s types of agile methods and distribution types to the new studied period from 2017-2021. The following analysis is based on Figure 4.7. Location (offshore/onshore): The combination used most frequently regarding the location was "Offshore-Scrum'. This scenario was reported in 22 cases from 2017-2021 compared to 34 cases in 2010-2016. This is significantly less which may be due to the increased usage of "Offshore-Scrum and Kanban" which was reported in 11 cases in 2017-2021. In addition, legal entity (outsourcing/insourcing): The combination "Outsourcing-Scrum" has been the most frequently adopted in 20 cases in total for the period 2017-2021. In comparison, in the years 2010-2016, "Insourcing-Scrum" was the most frequently reported in 20 cases compared to 8 cases from 2017-2021.

The following highlights the various distribution scenarios in GSD. Geographical distance (far, near): The combination "Far-Scrum" was practiced successfully the most (20 cases) for the period 2017-2021 compared to 28 cases for the years 2010-2016. Therefore, both studied periods revealed that Scrum was used the most in far geographical distances. Temporal distance (large, small): The combination "Large-

[13]Note that in the supplier to customer relationship in agile GSD, countries are represented as customers, and are either the clients for outsourcing business relationships or the main site for insourcing business relationships (Vallon et al., 2018).

Scrum" was found in 18 successful cases in 2017-2021 compared to 20 cases from 2010-2016. Therefore, the combination "Large-Scrum" was found the most in studies for the years 2010-2021. However, in 2010-2017 "Small-Scrum" also appeared in 20 cases compared to 6 cases from 2017-2021. Moreover, team distribution type (integrated, isolated): The combination "Integrated Teams-Scrum" was seen the most in 14 cases, (i.e., team members were spread across different sites and working together). Similarly, for the period 2010-2016, the most dominant distribution type was "Integrated Teams-Scrum" in 19 cases. Therefore, the new data reveals that this combination continues to be the most dominant team distribution type with GSD. Furthermore, the number of sites (2, 3, 4, and more): The combination of "Three Sites-Scrum" was practiced successfully the most (5 cases) in 2017-2021. However, for the period 2010-2016 "Two Sites-Scrum" was reported the most (20 cases). The years 2010-2016 reported on many cases that indicated the number of sites. However, this was not the case for the years 2017-2021, many studies did not explicitly state the details of sites, therefore, a total of 29 cases were "Unclear-Scrum".

Chapter 5

Discussion and Conclusion

The development of software across distributed locations with remote teams presents numerous challenges for organisations seeking to use agile methodologies to guide their work (Vallon et al., 2018). Such challenges include, for instance communication, time zones, distances, coordination, cultural differences, among others that are discussed throughout this thesis. The use of agile practices and communication media has been increasingly found in GSD literature to mitigate against some of the challenges, but also to help guide and solve challenges of work in agile software development in general (Vallon et al., 2018). Along with the challenges of GSD, there are numerous benefits to be gained in GSD such as globalization in the IT industry, whereby multicultural people and stakeholders around the world can work together on a global venue (Mahmood et al., 2022). In this study, the scope of the field was to focus on how agile practices have been successfully applied in GSD from 1999 to 2021 and, in particular, to identify the most widely adopted agile practices, distribution scenarios, and report on various challenges and communication tools adopted in these contexts. The purpose of this aim was to uncover any knowledge gaps and research opportunities in the field of agile GSD, and specifically compare the new data (2017-2021) to existing summary data (1999-2016) from Jalali and Wohlin (2012) and Vallon et al. (2018). This study is, therefore, a continuation of an ongoing research tradition in this field.

To address the stated aim, this study adopted a systematic literature search methodology and employed various bibliographic databases to investigate how agile software development practices have been successfully applied in globally distributed environments. In this chapter, a high-level overview of the main results of the study will be discussed, and core themes will be highlighted and compared to studies across different domains. Finally, the limitations of the study and relevant future work building on the findings will be discussed.

5.1 Discussion

The qualitative component of the results provided a rich discussion of key themes and patterns in the findings. Instead of repeating the findings in this study, this concluding chapter serves to integrate across these themes and consider them in the broader research domain and context to provide a more holistic view of agile practices in GSD. In this systematic literature review the majority of publications in GSD consisted of evaluation studies, and a significant portion of them used purely case study methods. In addition, most of the reviewed studies contributed to providing lessons learned and recommendations of practices in GSD. More importantly, this study found that, in many cases, team members faced similar challenges when collaborating remotely with communications media. Some of the common challenges identified include: communication, collaboration, coordination, time zones, and geographical and temporal distances. Furthermore, this study's data found that Scrum was the most frequently and successfully used agile method followed by the combination of Scrum and Kanban (Scrumban). This finding was evident across many studies, and the majority of the agile practices adopted were Scrum-orientated (i.e., sprints/iterations, backlog, sprint planning). Moreover, in the previously reviewed periods, not much variation or changes in terms of the adopted agile methods were found. However, towards the more recent period many combinations of mixed-method approaches were adopted, particularly when teams were located offshore. In most of the reviewed cases, members were distributed offshore and had to deal with far geographical, and large temporal distances.

In the following sections the seven main themes from both the comparative and qualitative analyses will be discussed and compared across the different studies to support the findings in this study.

5.1.1 *The dominance of Scrum Methods*

To answer RQ1b and RQ2b, in the earlier periods from 1999-2009, Jalali and Wohlin (2012) found that XP and Scrum methods were the most dominant methods.

However, the findings of the current study, and those of Vallon et al. (2018), indicate that Scrum was the most widely used method across studies. In this study, in the years 2017-2021, it was found that 38% of cases used purely Scrum methods on their own, and the decline of XP and other methods can be explained by the rise of mixed method approaches with Scrum, XP and other methods being used in combination. For example, in the present study, according to Figure 4.10, 29% of cases used agile methods in combination with Scrum. However, in Vallon et al. (2018)'s study, 60% of cases used Scrum methods. The findings in this study show less use of Scrum methods compared to the years 2010-2016, however, Scrum continues to remain the dominant method in situations where teams were distributed across several locations.

The dominance of Scrum methods is supported by a study conducted by Hoda et al. (2018) who refer to the latest state of agile survey which confirms that Scrum methods have increased their prominence as the most popular agile method in recent years in both distributed and co-located settings. In addition, Anwer et al. (2017) mentioned that Scrum is the most familiar and widely used agile software development model in the context of software development projects because it is easily adaptable and is able to accommodate for rapid application development needs. Similarly, a study conducted by Oomen et al. (2017) confirms that Scrum is one of the most popular agile methodologies in software development because it has been largely researched and used across organizations worldwide. Tech companies experience huge growth and success in Scrum methods because it is easy to implement, works in complex projects, and reduces time to market[14]. More importantly, given the current pandemic and rise of remote work, Scrum has become a popular way for organizations of all kinds to collaborate effectively, instill team transparency and increase accountability[15]. In the end, based on the present study findings and research abroad, Scrum is the most dominant not only in

[14]https://www.zeolearn.com/magazine/why-scrum
[15]https://www.sei.com/insights/article/your-roadmap-for-building-out-an-effective-data-driven-strategy-2/

remote working contexts but also in the context of other research areas of agile software development.

5.1.2 *Distribution of Scrum in agile GSD*

Scrum methods in agile GSD were the most dominant in various distribution fields. The broader term "distribution" in the context of this study is used to connote the different distribution settings (i.e., location, legal entity, sites, team distribution, geographical and temporal distances). There was a great variety of distribution scenarios in the years 2017-2021. In the earlier years, according to Jalali and Wohlin (2012), it was found that XP-offshore teams were the most popular distribution scenario. However, for Vallon et al. (2018)'s period (2010-2016) and the new period (2017-2021), it was observed that Scrum was the most dominant agile method used across all distribution scenarios. In this study, it was found that 22% of cases adopted Scrum methods in offshore environments (Offshore-Scrum). Moreover, in 20% of cases, Scrum was used when teams were faced with far geographical distances (Far-Scrum). In addition, in 20% of cases, Scrum methods were largely used when team members were insourced (Insourcing-Scrum). Therefore, based on this study's findings, it is concluded that (Offshore-Scrum) was the most dominant distribution scenario in GSD.

This finding is also supported by Seckin and Ovatman (2018), where their survey results from 24 individuals from 24 different projects concluded that Scrum methods were the most adopted in distributed teams (67% of cases) and followed core Scrum principles. Furthermore, Hron and Obwegeser (2018)'s study confirms that Scrum remains the dominant position in agile methods due to its widespread use, particularly in distributed settings where the Scrum team is divided into smaller teams across several locations. Moreover, Hron and Obwegeser (2018)'s study also confirms that Scrum software development principles are widely beneficial and applicable in various distribution scenarios. The present study concludes that Scrum-based distribution scenarios were the most prevalent and widespread in GSD due to the iterative and flexible nature of Scrum methods.

5.1.3 *Shifting away from XP to mixed methods*

To answer RQ3, according to Jalali and Wohlin (2012), many studies in the years 1999-2009 used purely XP in isolation to guide their software development procedures. However, in Vallon et al. (2018)'s study it was found that cases moved away from purely XP methods to adopting a combination of XP and Scrum methods. In the later periods (2010-2016) no cases used XP alone in GSD. According to the 15th annual agile report, less than 1% of respondents adopted XP practices with only 6% using XP in combination with Scrum, while 56% of respondents preferred mixed-method approaches in response to the Covid-19 pandemic[16]. Therefore, the shift away from purely XP methods to mixed methods is not only in GSD settings as highlighted in the present study, but across agile software development in general. Based on this study for the years 2017-2021, it was found that fewer studies used Scrum and XP compared to the years 1999-2016, and the shift was more to mixed methods and specifically the combination of Scrum and Kanban (Scrumban).

In this study it was observed that many cases adopted hybrid approaches, for example, XP, Scrum, Kanban, lean all working together in a project. Not only did this study find the increase usage of Scrum and Kanban, but the adoption of mixed method approaches in GSD which was not the case for the years 1999-2016. This increased use of mixed methods is evident in many companies across the globe. For instance, results from an international study, including over 400 projects, indicated that 52% of projects used hybrid approaches because it significantly increased stakeholder success and that hybrid is the leading project management approach (Gemino et al., 2021). Furthermore, it was found that hybrid methods had similar effectiveness compared to fully agile approaches (Gemino et al., 2021).

To further contextualize this theme, it was found in remote work for project management that many businesses are adopting hybrid or mixed method approaches to adjust to changing requirements (Adelakun et al., 2017). Therefore,

[16]https://digital.ai/resource-center/analyst-reports/state-of-agile-report

for many companies, mixed-method approaches are more flexible and draw strengths from both agile and traditional practices (Adelakun et al., 2017). Furthermore, Noll and Beechman (2019) analyzed data from the Helena survey[17] including 700 projects to assess how many projects combine agile methods, and it was found that the majority (66%) of the projects used mixed method approaches by combining different agile and traditional methods. Furthermore, according to the illustration *depicting the most popular agile methodologies* provided by Arora et al. (2021), Scrum is the most commonly used agile method in agile software development. In mixed-method software development using hybrids of Scrum is required for complex projects that require two different methodologies (i.e., Scrumban) and is becoming increasingly popular due to its flexible and continuous workflow (Arora et al., 2021). Therefore, based on this study's findings and research outside GSD, it can be concluded that the move to more mixed methods in GSD had nothing necessarily to do with GSD specifically – it is a general trend in software development.

5.1.4 *Changes in agile practices from earlier periods*

To answer RQ1b and RQ2b, in the earlier period (1999-2009), six practices out of the top ten were Scrum-based (e.g. standup meetings, sprint/iterations, sprint planning), in 2010-2016 eight of the top ten practices were affiliated with Scrum, and XP practices (e.g. standup meetings, backlog, sprint/iterations) were used less compared to Scrum-based practices. This indicated that the majority of studies in 2010-2016 used mixed approaches with Scrum. In this study, it was found for the years 2017-2021 that the combination of Scrum and Kanban was used in GSD studies and the majority of the agile practices were Scrum-orientated. Sprint/iterations being the most prevalant agile practice compared to standup meetings which were the most reported in the earlier periods (1999-2016).

[17]HELENA refers to one connected community and in this context the HELENA survey aims to investigate the use of mixed method approaches in software development.

In the context of project management, Scrum-based practices remain to be the most dominant because of its simplicity and interative/incremental development, and ability to be combined with other methodologies[18]. Furthermore, Masood et al. (2020) also confirm that Scrum-based practices are the most popular of agile practices and the most widely reported, and adapted project management framework. Through empirical evidence, this study can conclude that Scrum-based agile practices (i.e., sprints, product backlog, standup) remain the most dominant agile practices not only GSD but also across the entire spectrum of agile software development.

5.1.5 *Communication challenges and solutions*

In this study it was found that, due to the far geographical and large temporal distances that exist in GSD, many cases were faced with communication or collaboration issues (i.e., Alsahil, 2017; Batool, 2019). Unlike in co-located teams where members can interact face-to-face, globally distributed teams are unable to exercise face-to-face interaction and have to use tools or digital media to communicate and share ideas within the project (Henry et al., 2021). Therefore, when teams are globally distributed it can be challenging to conduct agile practices that are effective during face-to-face interaction such as daily standups and retrospectives. However, for example, teams are able to create a virtual environment who are able to interact and collaborate remotely but, rely heavily on social networking tools to perform virtual standup and retrospective meetings (Stray and Moe et al., 2020). Throughout the study, many cases involved team members that were distributed globally across several locations and experienced some level of communication challenges such as time zones, and cultural and temporal distances (i.e., Raith et al., 2017; Stray & Moe et al., 2020).

In order to deal with the communication challenges certain synchronous and asynchronous communication tools/media were used. Examples of such tools included Github, Slack, Skype, Teams, Zoom and various file sharing tools to

[18]https://content.intland.com

enable team members in global projects to communicate and collaborate effectively when teams were distributed across different locations (Merek et al., 2021). The aforementioned tools may increase team awareness, improve communication, and can mitigate against time zone and geographical distance challenges faced by globally distributed teams (Stray & Moe et al., 2020). However, in some situations teams may use personal direct calls between team members (i.e., Discord) and this can be perceived as creating separation between the teams because issues are discussed within sub-groups of the team (Rzhvskyi et al., 2020). Despite communication tools being able facilitate communication between team members, they are unable to solve time zone differences, therefore, organizing and planning meetings around different time zones, holidays and cultural events remains a challenge when members are globally distributed.

The study presented by Shameem et al. (2018) confirms that one of the main challenges faced by teams in GSD is communication, and the use of different communication channels to communicate, collaborate and support agile practices such as stand-up meetings, retrospectives, and planning. Given the nature of global software development, and temporal distances, face-to-face communication is seldomly employed in GSD, therefore, communication channels such as video conferencing, audio conferencing, and email are used to ensure open and multidirectional interaction (Ahmed et al., 2018). However, according to Rzhvskyi et al. (2020), agile teams agreed there were many communication difficulties and the tools used represented a conditioning factor for whole project. In some cases, agile practices (i.e., standup meetings, retrospectives) had to be adapted. In relation to this study findings and other research in GSD, it can be concluded that team members in distributed settings are faced various communication challenges and require to use information and communication technology to interact and collaborate abroad.

5.1.6 *Challenge comparison between distribution in one country/region vs. across multiple countries*

Research reviewed in this study suggests that when teams are isolated and distributed at a single location/country they are ultimately faced with fewer communication, cultural and language challenges compared to teams that are dispersed globally across several locations. The reason being is teams that are globally distributed have to be prepared to work with different cultures, languages, time zone differences, and different levels of competency. Multiple case studies (i.e., Smite et al., 2020; Beecham et al., 2021) found that teams are faced with temporal, geographical and cultural challenges when distributed across multiple countries. For example, if members are distributed across India and the UK, there are large time zone differences, cultural and language issues to deal with, and the level of understanding will differ between team members. In cases where teams were distributed at a single location (i.e., do Santos et al. 2018; Khan et al. 2021) did not have to deal with time zones, temporal, and geographical challenges.

To further support this finding, Bjorn et al. (2017) suggest that team members working in one location face fewer time zone challenges and cultural differences because the members are adapted to that particular region. However, when members are distributed across different countries, many cultural and time zone challenges are presented. Furthermore, members working in one country work under different transnational circumstances compared to those located across different countries (Bjorn et al., 2017). Employees in different countries have certain ways of using agile methodologies, tools, technology, and ways of working that shape the translocality of the workplace. In addition, co-located members may experience informal communication, while workers distributed across different locations are isolated and possibly excluded from the knowledge network that co-located teams are embedded with (Deshpande et al., 2016). Therefore, based on this study's findings and other GSD research it can be concluded that teams over multiple locations and those distributed in one location deal with different challenges. The concept of having all members in one single location can lead to

inflexibility which is difficult to maintain, particularly since the Covid-19 pandemic where much work has been done remotely[19]. Comparatively, challenges in distributed settings such as different times, distances, and feeling of disconnect are experienced which can be mitigated against through use of tools and media (Bjorn et al. (2017).

5.1.7 *A stable and maturing agile GSD field*

Vallon et al. (2018) argued that agile software development in globally distributed environments is a growing and maturing area because of the shift away from case studies and the increasing number of reviews. The results of the present study (2017-2021) support this trend and show that more studies were literature reviews (22% of cases) with fewer studies adopting case study analysis (18% of cases). Some of the key changes in this study that indicated that GSD is a maturing field was the shift from XP methods to Scrum and mixed methods. Moreover, in the earlier years (1999-2009) the most frequent publication type was experience reports, such studies report in an anecdotal way and focus less on reporting key findings from primary data. However, for the later period from 2010-2016, it was clear that the maturing research field shifted from experience reports to evaluation studies. The new studied period 2017-2021, supports the findings for the years 2010-2016 because the newly collected data revealed that evaluation studies remain the most dominant publication type. However, not only did evaluation studies remain the most frequent publication type, but they also showed means of growth for the years 2017-2021. Specifically, an increase of 8% was identified compared to the earlier periods which may indicate signs of growth in systematic literature reviews in GSD.

Another important observation was that more studies contributed frameworks and tools compared to case study analysis. This may be indicative of the research field reaching maturity because more studies are proposing theories, solutions,

[19]https://techbeacon.com/app-dev-testing/distributed-vs-colocated-agile-teams-pros-cons

recommendations, and through the developing process are able to provide complete and working software in agile GSD. Therefore, all of these significant changes in later periods prove that GSD is a maturing research field, however, few evidence in this study suggests that the field is growing, but rather that the number of publications is constant in GSD.

Vallon et al. (2018)'s study confirms that agile GSD is a maturing field with higher quality contributions and a great variety of methods from 2010-2016. To further support that GSD is stable and maturing research field, Camara et al. (2020)'s study observed that the number of publications became constant since 2011, however, a growing number of evaluation studies was found which may indicate growth in systematic literature reviews. This finding by Camara et al. (2020) study is almost identical to that presented in this study and suggests that the field is reaching maturity. According to Ebert et al. (2016) study, GSD is an evolving research area and will eventually reach some standard software method, and only those who successfully manage their distributed projects will succeed in the future. Based on the data discussed in Vallon et al. (2018) and the findings of the present study, it can be concluded that the GSD field is more stable than growing but show immediate signs of maturity.

5.2 Limitations

Although this systematic literature review provides valuable insight into software development in distributed and remote contexts and extends Vallon et al. (2018)'s earlier work that primarily relied on data from Jalali and Wohlin (2012), it is not without limitations. The studies selected in this systematic literature search are based primarily on the search terms used such as "global software development", "remote work". Therefore, the use of more traditional terms like "lean" and "outsourcing" could have expanded the search space. Nonetheless, the use of search terms, while a potential limiting factor, follow norms adopted in previous research. Moreover, lack of access to certain databases such as AIS library was a limitation, however, there is substantial overlap and coverage with Scopus.

A further limitation concerns the period for which the new data was collected. The studied period is only 2017-2021, a longer period of analysis could bring more GSD studies and Covid-related research which would have been included for review. Another limitation, the systematic review was a continuation of the Vallon et al. (2018) study, and a similar research design was adopted. Furthermore, many of Vallon et al. (2018) limitations are a potential threat to this study and may affect the findings of this study. Also, the summary data was used from Jalali and Wohlin (2012) and Vallon et al. (2018). Therefore, the accuracy and quality of existing data cannot be verified. Moreover, one of the major limiting factors in this review was that cases were mostly defined as successful. It is hard to know if this means that all cases are successful or if there is a publication bias and only successful cases are published. Irrespective of the cause, this may have limited the coverage of the review. Lastly, as was also the case in Vallon et al. (2018), some of the full-text versions in studies were not readily available for review.

5.3 Future Directions and Recommendations

The following are some possible future opportunities to improve and grow research within this domain. More studies in this field should report on both success and failure cases because in the present study it was found that most cases reported on success and seldomly reported on failure. Reporting more on failure cases will aid future research in the agile GSD field to produce new research on the successful and failed implementation of agile practices in agile GSD. Along with the development of criteria for describing what, why and how an agile implementation failed or succeeded, there is a need to report on the full empirical characteristics of studies to help future studies extract contextual information. Moreover, there is a need to investigate more agile software development practices (*i.e., mixed methods*) because agile methods like Scrum, and Scrum in combination with agile methods have become a popular way for companies to collaborate effectively in remote working contexts. The new data reveals, the combination of Scrum and Kanban is growing and therefore, there is a need to investigate into more Kanban-related practices. Furthermore, another future direction can be investigating failure cases

because studies did not report on cases that failed and describe under what conditions did agile practices fail in remote contexts. Specifically, qualitative research on developers' actual experiences with agile GSD. Also, future studies should specifically focus on the impact of the Covid-19 pandemic on GSD practices. Lastly, another recommendation for future research in this domain is to produce a large-scale comparative review in 5 years to continue the trend from Jalali and Wohlin (2012), Vallon et al. (2018), and this study.

5.4 Conclusion

This systematic review was a continuation of Jalali and Wohlin (2012), and Vallon et al. (2018)'s studies and reviewed the usage of agile practices in GSD from 1999 to 2021. This was executed by analyzing both the 1999-2021 period as a whole as well as comparing the periods studied by Jalali and Wohlin (2012) and Vallon et al. (2018) against the new studied period of 2017-2021. The main research contribution of this study is in reporting the current state of agile practices in GSD for the years 1999-2021, where this study almost exclusively found successful cases. It was found in these cases that sprint/iterations, standup meetings, and backlogs were the most frequently adopted agile practices in GSD settings. The GSD field had many changes in terms of publication types and research methods which were indicative of a growing and maturing research field. However, more importantly, the study found numerous changes and new combinations of agile methods, and practices to deal with the challenges of GSD. Furthermore, this study has highlighted the importance of ICTs to facilitate agile practices in the context of remote work, and how it can bring people together around the world to work on a global project. In the end, this study contributed by continuing the research conducted by the Vallon et al. (2018) and provided a recent update on agile GSD which is an important part of a much-needed effort to deal with agile remote work challenges in the future.

Appendix A

Electronic Database Search Strings

A.1 Scopus

TITLE-ABS-KEY (((agile OR scrum OR "extreme programming" OR "pair programming" OR "lean development" OR "lean software development") AND ("global distributed 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 "remote team" OR "virtual team" OR offshore OR outsource))) AND PUBYEAR > 2016 AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cr") OR LIMIT-TO (DOCTYPE , "ch") OR LIMIT-TO (DOCTYPE , "re"))

Results: *n* = 227 (20/08/2021)

A.2 IEEE Xplore

("Abstract":agile OR "Abstract":scrum OR "Abstract":"extreme programming" OR "Abstract":"pair programming" OR "Abstract":"lean development" OR "Abstract":"lean software development")
AND
("Abstract":"distributed software development" OR "Abstract": "global software development" OR "Abstract": "distributed software engineering" OR "Abstract": "global distributed software engineering" OR "Abstract":gse OR "Abstract":gsd OR "Abstract":"distributed team" OR "Abstract":"global team" OR "Abstract":"dispersed team" OR "Abstract":"remote team" OR "Abstract":"virtual team" OR "Abstract":offshore OR "Abstract":outsource)

Results: *n* = 57 (20/08/2021)

A.3 ACM

Brief Query:
Abstract:(agile OR scrum OR "extreme programming" OR "pair programming" OR "lean development" OR "lean software development") AND Abstract:("global distributed 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 "remote team" OR "virtual team" OR offshore OR outsource)

Full Query:
"query": { Abstract:(agile OR scrum OR "extreme programming" OR "pair programming" OR "lean development" OR "lean software development") AND Abstract:("global distributed 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 "remote team" OR "virtual team" OR offshore OR outsource) }
"filter": { Publication Date: (01/01/2017 TO *), ACM Content: DL}

Results: *n* = 30 (20/08/2021)

A.4 Compendex

Full Query:
(((agile OR scrum OR "extreme programming" OR "pair programming" OR "lean development" OR "lean software development") WN KY) AND (("global distributed 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 "remote team" OR "virtual team" OR offshore OR outsource) WN KY)) AND (English WN LA)

Filters applied: Databases = Compendex; Date > 2016

((((agile OR scrum OR "extreme programming" OR "pair programming" OR "lean development" OR "lean software development") WN KY) AND (("global distributed 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 "remote team" OR "virtual team" OR offshore OR outsource) WN KY)) AND (English WN LA)) + (({ca} OR {ja} OR {cp} OR {ch} OR {ip}) WN DT) AND ({english} WN LA)

Results: *n* = 145 (20/08/2021)

Appendix B

Data Extraction Form

General

- Include (yes, no, maybe)
- Exclusion comment
- Identifier
- Title
- Databases (comma-separated)
- Authors' Names (comma-separated)
- Authors' Affiliations (comma-separated: specific university/R&D or "INDUSTRY")
- Authors' Countries (comma-separated)
- Publication Year
- Target (conference or journal name)
- Timestamp of Data Extraction and Researcher's Name

Research

- Type (solution, validation, evaluation, philosophical, experience, opinion)
- Method (qualitative, quantitative, mixed approach)
- Sub-Method (single-case study, multiple-case study, interviews, etc.)
- Means of Analysis (grounded theory, statistical, etc.)

Empirical

- Empirical (yes, no, unclear)
- Project Size (small, medium, large, unclear)
- Project Duration (short, medium, large, unclear)
- Participants (industry, students, unclear)

- Knowledge Area (requirement, design, construction, testing, SE management, SE process, maintenance, tools & methods)
- Application Domain (e.g. enterprise software, etc.)
- Successful (yes, no, unclear)
- Failure (yes, no, unclear)

Remote Work

- Global Software Development (yes, no, unclear)
- Location (offshore, onshore, unclear)
- Legal entity (outsourcing, insourcing, unclear)
- Geographical Distance (far, near, unclear)
- Temporal Distance (large, small, unclear)
- Team Distribution Type (co-located, isolated, unclear)
- Number of sites (0 for unclear, otherwise > = 1)
- Supplier Countries (comma-separated)
- Customer Countries (comma-separated)

Agile

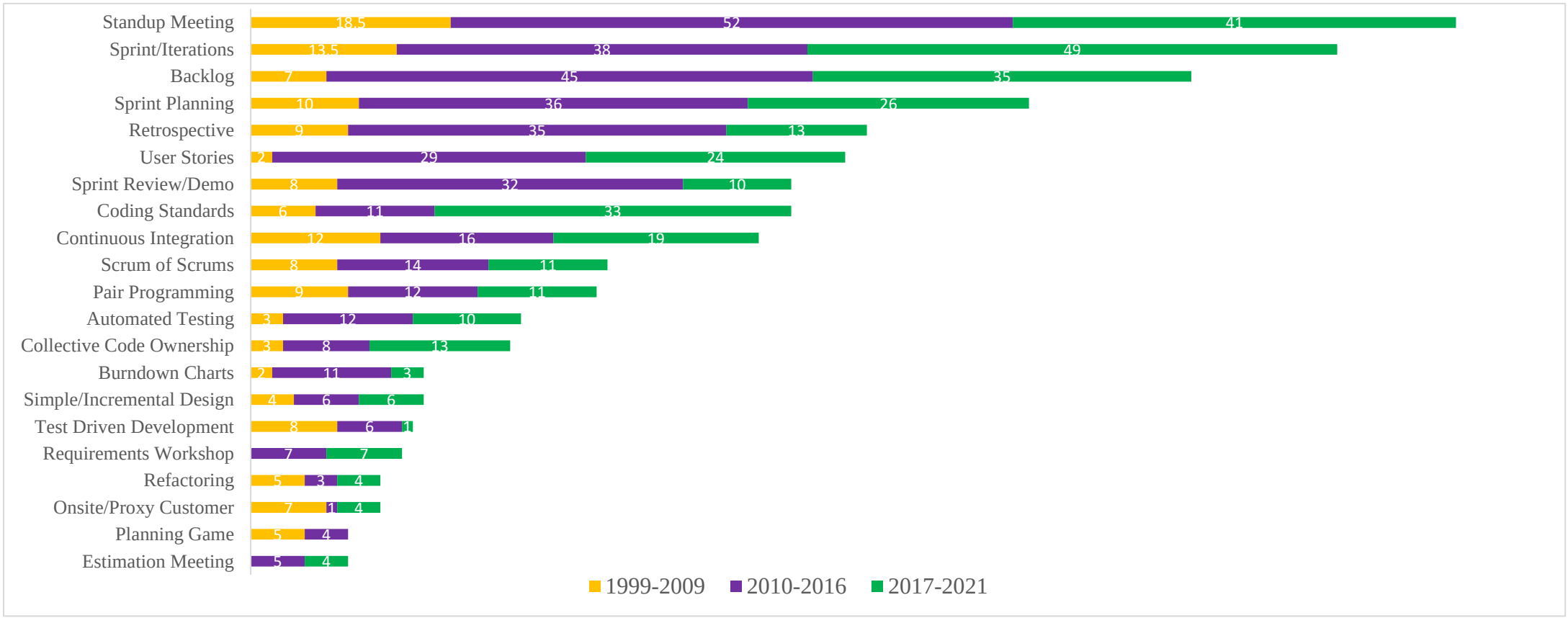
- Agile Method (Scrum, XP, Lean, Kanban, unclear)
- Agility Level (not all teams, all teams, organization, unclear)
- Agile Practices (comma-separated)

Result

- Contributions of the Study (lessons learned, tool development, framework, model, etc.)
- Summary Comment by Research

Appendix C

Overview of agile practices for the years 1999-2021



Appendix D

Bibliographic Details of Reports Systematically Reviewed

This appendix presents the bibliographic details for the full-text reports considered in the systematic review. First, the details for those reports included in the final sample are presented. Next, following this, the bibliographic details for this excluded from the review are provided. In chapter 3, the PRISMA flowchart represents the criteria for which a study was excluded. Reports indicated by an (*) are part of the stage 2 analysis.

D.1 Reports Included in the Review for 2017-2021

1. *Afshari, M., & Javdani Gandomani, T. (2021). Quality of agile adoption in global software development: An assessment model. *Indonesian Journal of Electrical Engineering and Computer Science*, *21*(1), 367. https://doi.org/10.11591/ijeecs.v21.i1.pp367-376

2. Aggarwal, A. K., & Mani, V. S. (2019). Using Product Line Engineering in a Globally Distributed Agile Development Team to Shorten Release Cycles Effectively. *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, 58–61. https://doi.org/10.1109/ICGSE.2019.00023

3. Ahmad, A., Hill, R., Lu, J., Mccluskey, L., & Wade, S. (2020). An Analysis on Scrum Methodology in Global Software Development – GSD. *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, 1807–1812. https://doi.org/10.1109/CSCI51800.2020.0033

APPENDIX D. BIBLIOGRAPHIC DETAILS OF REPORTS SYSTEMATICALLY REVIEWED

4. Ahmad, M. O., Lenarduzzi, V., Oivo, M., & Taibi, D. (2018). *Lessons Learned on Communication Channels and Practices in Agile Software Development*. 929–938. https://doi.org/10.15439/2018F7

5. Aizaz, F., Khan, S. U. R., Khan, J. A., Inayat-Ur-Rehman, & Akhunzada, A. (2021). An Empirical Investigation of Factors Causing Scope Creep in Agile Global Software Development Context: A Conceptual Model for Project Managers. *IEEE Access*, *9*, 109166–109195. https://doi.org/10.1109/ACCESS.2021.3100779

6. *Akbar, M. A., Khan, A. A., Khan, A. W., & Mahmood, S. (2020). Requirement change management challenges in GSD: An analytical hierarchy process approach. *Journal of Software: Evolution and Process*, *32*(7). https://doi.org/10.1002/smr.2246

7. Akbar, M. A., Mahmood, S., Khan, A. A., AlSanad, A., & Gumaei, A. (2020). Prioritizing Management Success Factors in Offshore Software Development. *Arabian Journal for Science and Engineering*, *45*(12), 10163–10184. https://doi.org/10.1007/s13369-020-04607-2

8. Akbar, M. A., Sang, J., Nasrullah, Khan, A. A., Mahmood, S., Qadri, S. F., Hu, H., & Xiang, H. (2019). Success factors influencing requirements change management process in global software development. *Journal of Computer Languages*, *51*, 112–130. https://doi.org/10.1016/j.cola.2018.12.005

9. *Akbar, R., Khan, A. R., & Adnan, K. (2020). Software Development Process Evolution in Malaysian Companies. *Advances in Intelligent Systems and Computing*, *1042*, 139–150. https://doi.org/10.1007/978-981-32-9949-8_10

10. Alsahli, A., Khan, H., & Alyahya, S. (2017). *Agile Development Overcomes GSD Challenges: A Systematic Literature Review*. 13.

11. *Alsaqaf, W., Daneva, M., & Wieringa, R. (2019). Quality requirements challenges in the context of large-scale distributed agile: An empirical study. *Information and Software Technology*, *110*, 39–55. https://doi.org/10.1016/j.infsof.2019.01.009

12. *Al-Zaidi, A., & Qureshi, R. (2017). *Global Software Development Geographical Distance Communication Challenges*. *14*(2), 8.

13. Alzoubi, Y. I., & Gill, A. Q. (2020). An Empirical Investigation of Geographically Distributed Agile Development: The Agile Enterprise Architecture is a Communication Enabler. *IEEE Access*, *8*, 80269–80289. https://doi.org/10.1109/ACCESS.2020.2990389

14. *Arbain, A. F., Rafeek, M. A., Podari, Z., & Mohd Foozy, C. F. (2021). Measuring Risk Mitigation Techniques in Agile Global Software Development. *Lecture Notes on Data Engineering and Communications Technologies*, *72*, 1225–1232. https://doi.org/10.1007/978-3-030-70713-2_109

15. *Arumugam, C., Vaidayanthan, S., & Karuppuchamy, H. (2018). Global software development: Key Performance measures of team in a SCRUM based agile environment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10963 LNCS*, 672–682. https://doi.org/10.1007/978-3-319-95171-3_53

16. *Arumugam, C., & Vaidyanathan, S. (2019). Agile team measurement to review the performance in global software development. In *Agile team measurement to review the performance in global software development*. https://doi.org/10.4018/978-1-5225-9659-2.ch005

17. *Ashmore, S., Townsend, A., DeMarie, S., & Mennecke, B. (2018). An exploratory examination of modes of interaction and work in waterfall and

agile teams. *International Journal of Agile Systems and Management*, *11*(1), 67–102. https://doi.org/10.1504/ijasm.2018.091361

18. Aslam, A., Ahmad, N., Saba, T., Almazyad, A. S., Rehman, A., Anjum, A., & Khan, A. (2017). Decision Support System for Risk Assessment and Management Strategies in Distributed Software Development. *IEEE Access*, *5*, 20349–20373. https://doi.org/10.1109/ACCESS.2017.2757605

19. Aslam, W., & Ijaz, F. (2018). A Quantitative Framework for Task Allocation in Distributed Agile Software Development. *IEEE Access*, *6*, 15380–15390. https://doi.org/10.1109/ACCESS.2018.2803685

20. *Awar, K. B., Sameem, M. S. I., & Hafeez, Y. (2017). A model for applying Agile practices in Distributed environment: A case of local software industry. *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*, 228–232. https://doi.org/10.1109/C-CODE.2017.7918933

21. *Aziz, T., Haq, E., Tariq, S., & Batool, H. (2018). Influence of Project Management in Requirement Engineering Process for Global Software Development. *International Journal of Computer Applications*, *181*(28), 28–35. https://doi.org/10.5120/ijca2018918132

22. *Banijamali, A., Dawadi, R., Ahmad, M. O., Similä, J., Oivo, M., & Liukkunen, K. (2017). An Empirical Study on the Impact of Scrumban on Geographically Distributed Software Development: *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development*, 567–577. https://doi.org/10.5220/0005686405670577

23. *Banijamali, A., Dawadi, R., Ahmad, M. O., Similä, J., Oivo, M., & Liukkunen, K. (2017). Empirical Investigation of Scrumban in Global Software Development. In S. Hammoudi, L. F. Pires, B. Selic, & P. Desfray (Eds.), *Model-Driven Engineering and Software Development* (Vol. 692,

pp. 229–248). Springer International Publishing. https://doi.org/10.1007/978-3-319-66302-9_12

24. *Baschin, J., Inkermann, D., & Vietor, T. (2019). Agile Process Engineering to support Collaborative Design. *Procedia CIRP*, *84*, 1035–1040. https://doi.org/10.1016/j.procir.2019.05.010

25. Batool, A., Chowdhury, M., & Chowdhury, A. (2019). *A Survey of Key Challenges of Adopting Agile in Global Software Development: A Case Study with Malaysia Perspective*. 7.

26. *Beecham, S., Clear, T., Lal, R., & Noll, J. (2021). Do scaling agile frameworks address global software development risks? An empirical study. *Journal of Systems and Software*, *171*, 110823. https://doi.org/10.1016/j.jss.2020.110823

27. *Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2018). Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. *IEEE Transactions on Software Engineering*, *44*(10), 932–950. https://doi.org/10.1109/TSE.2017.2730870

28. *Bjørn, P., Søderberg, A.-M., & Krishna, S. (2019). Translocality in Global Software Development: The Dark Side of Global Agile. *Human–Computer Interaction*, *34*(2), 174–203. https://doi.org/10.1080/07370024.2017.1398092

29. *Borrego, G., Moran, A. L., & Palacio, R. (2017). Preliminary Evaluation of a Tag-Based Knowledge Condensation Tool in Agile and Distributed Teams. *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 51–55. https://doi.org/10.1109/ICGSE.2017.14

30. *Borrego, G., Morán, A. L., Palacio, R. R., Vizcaíno, A., & García, F. O. (2019). Towards a reduction in architectural knowledge vaporization during

agile global software development. *Information and Software Technology*, *112*, 68–82. https://doi.org/10.1016/j.infsof.2019.04.008

31. *Borrego, G., Moran, A. L., Palacio, R., & Rodriguez, O. M. (2016a). Findings on AGSD Architectural Knowledge Sharing. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 193–194. https://doi.org/10.1109/ICGSE.2016.38

32. *Borrego, G., Moran, A. L., Palacio, R., & Rodriguez, O. M. (2016b). Understanding Architectural Knowledge Sharing in AGSD Teams: An Empirical Study. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 109–118. https://doi.org/10.1109/ICGSE.2016.29

33. *Bosnic, I., & Cavrak, I. (2019). Project Work Division in Agile Distributed Student Teams—Who Develops What? *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, 162–171. https://doi.org/10.1109/ICGSE.2019.00037

34. Calefato, F., & Ebert, C. (2019). Agile Collaboration for Distributed Teams [Software Technology]. *IEEE Software*, *36*(1), 72–78. https://doi.org/10.1109/MS.2018.2874668

35. Camara, R., Alves, A., Monte, I., & Marinho, M. (2020). Agile Global Software Development: A Systematic Literature Review. *Proceedings of the 34th Brazilian Symposium on Software Engineering*, 31–40. https://doi.org/10.1145/3422392.3422411

36. Castelle, K., & Dario, E. (2019). *Knowledge Exchange Mechanisms in Agile Environments for Globally Outsourced Teams*. 7.

37. *Chilito, P., Viveros, D., Pardo, C., & Pino, F. J. (2018). Scrum+: An agile guide for the global software development (GSD) multi-model project management. *2018 IEEE Colombian Conference on Communications and*

Computing (COLCOM), 1–6. https://doi.org/10.1109/ColComCon.2018.8466710

38. Christopher, L., & de Vries, M. (2020). SELECTING A SCALED AGILE APPROACH FOR A FIN-TECH COMPANY. *South African Journal of Industrial Engineering*, *31*(3). https://doi.org/10.7166/31-3-2432

39. *Counsell, S., Modi, S., & Abbott, P. (2020). Themes and Difficulties in Distributed Agile Email Activity: A Qualititative Team-Based Study. *Proceedings of the Evaluation and Assessment in Software Engineering*, 288–292. https://doi.org/10.1145/3383219.3383250

40. *Cruz, A. F. da, Godoy, C. P., Santos, L. M. dos, Marinho, L. F., Jardim, M. S., Silva, E. P. da, Pahins, C. A., Fonseca, P., & Giuntini, F. T. (2020). Blueprint Model: An Agile-Oriented Methodology for Tackling Global Software Development Challenges. *Advances in Science, Technology and Engineering Systems Journal*, *5*(6), 353–362. https://doi.org/10.25046/aj050643

41. *Cruzes, D. S., Moe, N. B., & Dyba, T. (2016). Communication between Developers and Testers in Distributed Continuous Agile Testing. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 59–68. https://doi.org/10.1109/ICGSE.2016.27

42. Cultural Issues in Offshore Teams: A Categorization based on Existing Studies. (2019). *KSII Transactions on Internet and Information Systems*, *13*(3). https://doi.org/10.3837/tiis.2019.03.014

43. Cuong, L. G., Hung, P. D., Bach, N. L., & Tung, T. D. (2019). Risk Management for Agile Projects in Offshore Vietnam. *Proceedings of the Tenth International Symposium on Information and Communication Technology - SoICT 2019*, 377–384. https://doi.org/10.1145/3368926.3369718

44. Cuong, L. G., Hung, P. D., & Vinh, B. T. (2018). Migrating Vietnam Offshore into Agile. *Proceedings of the Ninth International Symposium on Information and Communication Technology - SoICT 2018*, 329–336. https://doi.org/10.1145/3287921.3287924

45. Dayala, P. P., & Rajkumarb, M. (2021) Agile practices in global software development: A topic modelling approach.

46. *Dafoulas, G., Maia, C., Ali, A., Augusto, J. C., & Lopez-Cabrera, V. (2017). Understanding Collaboration in Global Software Engineering (GSE) Teams with the Use of Sensors: Introducing a Multi-sensor Setting for Observing Social and Human Aspects in Project Management. *2017 International Conference on Intelligent Environments (IE)*, 114–121. https://doi.org/10.1109/IE.2017.40

47. *dos Santos, L. S., L'Erario, A., Pagotto, T., Camilo, J. R. M., Oliveira, F. S., & Fabri, J. A. (2018). A scrum-based process to distributed projects in multidisciplinary teams: A case study. *Proceedings of the 13th International Conference on Global Software Engineering*, 133–134. https://doi.org/10.1145/3196369.3196380

48. *Ebert, C. (2018). Managing software products in a global context. *Proceedings of the 13th International Conference on Global Software Engineering*, 69–76. https://doi.org/10.1145/3196369.3196371

49. *Esteki, M., Javdani Gandomani, T., & Khosravi Farsani, H. (2020). A risk management framework for distributed scrum using PRINCE2 methodology. *Bulletin of Electrical Engineering and Informatics*, *9*(3), 1299–1310. https://doi.org/10.11591/eei.v9i3.1905

50. *Feitoza Gonçalves, W., de Farias Junior, I., de Paulo Alves, R. K., Saraiva Barbosa, P. L., Ribeiro Parente Cortez, H., Bezerra de Oliveira, I., Mendonça Teixeira, M., & Leitão Júnior, N. (2017). Using Agile Methods

in Distributed Software Development Environments. In T. Silva da Silva, B. Estácio, J. Kroll, & R. Mantovani Fontana (Eds.), *Agile Methods* (Vol. 680, pp. 16–27). Springer International Publishing. https://doi.org/10.1007/978-3-319-55907-0_2

51. *Gervigny, M. L. I., & Nagowah, S. D. (2017). Knowledge sharing for agile distributed teams: A case study of Mauritius. *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, 413–419. https://doi.org/10.1109/ICTUS.2017.8286043

52. *Godoy, C. P., Cruz, A. F., Silva, E. P., Santos, L. M., Zerbini, R. S., & Pahins, C. A. L. (2019). Blueprint Model: A new Approach to Scrum Agile Methodology. *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, 95–99. https://doi.org/10.1109/ICGSE.2019.00014

53. *Guillot, I., Paulmani, G., Kumar, V., & Fraser, S. N. (2017). Case studies of industry-academia research collaborations for software development with Agile. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10391 LNCS*, 196–212. https://doi.org/10.1007/978-3-319-63874-4_15

54. *Gupta, R. K., & Anand, T. (2017). Knowledge Transfer for Global Roles in GSE. *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 81–85. https://doi.org/10.1109/ICGSE.2017.2

55. Hafeez, Y., Asghar, S., Arif, B., & Ali, S. (2021). Role of situational method engineering to improve visual information systems in agile distributed environment. *Multimedia Tools and Applications*, *80*(6), 8881–8908. https://doi.org/10.1007/s11042-020-09896-1

56. Hidayati, A., Budiardjo, E. K., & Purwandari, B. (2020). Hard and Soft Skills for Scrum Global Software Development Teams. *Proceedings of the 3rd International Conference on Software Engineering and Information Management*, 110–114. https://doi.org/10.1145/3378936.3378966

57. Hoda, R., Salleh, N., Grundy, J., & Tee, H. M. (2017). Systematic literature reviews in agile software development: A tertiary study. *Information and Software Technology*, *85*, 60–70. https://doi.org/10.1016/j.infsof.2017.01.007

58. *Hossain, S. S. (2019). Challenges and Mitigation Strategies in Reusing Requirements in Large-Scale Distributed Agile Software Development: A Survey Result. *Advances in Intelligent Systems and Computing*, *998*, 920–935. https://doi.org/10.1007/978-3-030-22868-2_63

59. *Huber, T. L., Winkler, M. A. E., Dibbern, J., & Brown, C. V. (2020). The use of prototypes to bridge knowledge boundaries in agile software development. *Information Systems Journal*, *30*(2), 270–294. https://doi.org/10.1111/isj.12261

60. Ilyas, M., & Khan, S. U. (2017). Software integration in global software development: Challenges for GSD vendors. *Journal of Software: Evolution and Process*, *29*(8), e1875. https://doi.org/10.1002/smr.1875

61. *Jeeva Padmini, K. V., Kankanamge, P. S., Bandara, H. M. N. D., & Perera, G. I. U. S. (2018). Challenges Faced by Agile Testers: A Case Study. *2018 Moratuwa Engineering Research Conference (MERCon)*, 431–436. https://doi.org/10.1109/MERCon.2018.8421968

62. Jha, M. M., Vilardell, R. M. F., & Narayan, J. (2016). Scaling Agile Scrum Software Development: Providing Agility and Quality to Platform Development by Reducing Time to Market. *2016 IEEE 11th International*

Conference on Global Software Engineering (ICGSE), 84–88. https://doi.org/10.1109/ICGSE.2016.24

63. Kalenda, M., Hyna, P., & Rossi, B. (2018). Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*, *30*(10), e1954. https://doi.org/10.1002/smr.1954

64. *Kahya, M. D., & Seneler, Ç. (2018). Geographical Distance Challenges in Distributed Agile Software Development: Case Study of a Global Company. *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, 78–83. https://doi.org/10.1109/UBMK.2018.8566591

65. *Kamal, T., Zhang, Q., & Akbar, M. A. (2020). Toward successful agile requirements change management process in global software development: A client–vendor analysis. *IET Software*, *14*(3), 265–274. https://doi.org/10.1049/iet-sen.2019.0128

66. Kamal, T., Zhang, Q., Akbar, M. A., Shafiq, M., Gumaei, A., & Alsanad, A. (2020). Identification and Prioritization of Agile Requirements Change Management Success Factors in the Domain of Global Software Development. *IEEE Access*, *8*, 44714–44726. https://doi.org/10.1109/ACCESS.2020.2976723

67. Kausar, M., & Al-Yasiri, A. (2017). Using distributed agile patterns for supporting the requirements engineering process. In *Requirements Engineering for Service and Cloud Computing*. https://doi.org/10.1007/978-3-319-51310-2_13

68. Khan, A. A., Keung, J., Niazi, M., Hussain, S., & Shameem, M. (2019). GSEPIM: A roadmap for software process assessment and improvement in

the domain of global software development. *Journal of Software: Evolution and Process*, *31*(1), e1988. https://doi.org/10.1002/smr.1988

69. *Khan, A. A., Shameem, M., Nadeem, M., & Akbar, M. A. (2021). Agile trends in Chinese global software development industry: Fuzzy AHP based conceptual mapping. *Applied Soft Computing*, *102*, 107090. https://doi.org/10.1016/j.asoc.2021.107090

70. *Khan, H. U., Niazi, M., El-Attar, M., Ikram, N., Khan, S. U., & Gill, A. Q. (2021). Empirical Investigation of Critical Requirements Engineering Practices for Global Software Development. *IEEE Access*, *9*, 93593–93613. https://doi.org/10.1109/ACCESS.2021.3092679

71. *Khmelevsky, Y., Li, X., & Madnick, S. (2017). Software development using agile and scrum in distributed teams. *2017 Annual IEEE International Systems Conference (SysCon)*, 1–4. https://doi.org/10.1109/SYSCON.2017.7934766

72. *Kiely, G., Kiely, J., & Nolan, C. (2017). Scaling Agile Methods to Process Improvement Projects: A Global Virtual Team Case Study. *AMCIS*.

73. Kuhrmann, M., Diebold, P., Munch, J., & Tell, P. (2016). How Does Software Process Improvement Address Global Software Engineering? *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 89–98. https://doi.org/10.1109/ICGSE.2016.10

74. *Kuivalainen, J., Kunttu, I., & Kohtamäki, M. (2020). *Agile Product Development Practices for Coping with a Learning Paradox in R&D Offshore Units*. 9.

75. Kumar, M. R., & Mani, V. S. (2018). Transitioning from plan-driven to lean in a global software engineering organization: A practice-centric view.

Proceedings of the 13th International Conference on Global Software Engineering, 1–5. https://doi.org/10.1145/3196369.3196395

76. *Lauren, B. S. (2017). Mapping the workspace of a globally distributed "agile" team. *Mapping the Workspace of a Globally Distributed "Agile" Team*, *1–2*. https://doi.org/10.4018/978-1-5225-1918-8.ch009

77. *Lal, R., & Clear, T. (2018). Enhancing Product and Service Capability through Scaling Agility in a Global Software Vendor Environment. *Proceedings of the 13th International Conference on Global Software Engineering*, 59–68. https://doi.org/10.1145/3196369.3196378

78. *Lautert, T., Neto, A. G. S. S., & Kozievitch, N. P. (2019). A Survey on Agile Practices and Challenges of a Global Software Development Team. In P. Meirelles, M. A. Nelson, & C. Rocha (Eds.), *Agile Methods* (Vol. 1106, pp. 128–143). Springer International Publishing. https://doi.org/10.1007/978-3-030-36701-5_11

79. Lous, P., Kuhrmann, M., & Tell, P. (2017). Is Scrum Fit for Global Software Engineering? *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 1–10. https://doi.org/10.1109/ICGSE.2017.13

80. *Lous, P., Tell, P., Michelsen, C. B., Dittrich, Y., & Ebdrup, A. (2018). From Scrum to Agile: A journey to tackle the challenges of distributed development in an Agile team. *Proceedings of the 2018 International Conference on Software and System Process*, 11–20. https://doi.org/10.1145/3202710.3203149

81. *Lous, P., Tell, P., Michelsen, C. B., Dittrich, Y., Kuhrmann, M., & Ebdrup, A. (2018). Virtual by design: How a work environment can support agile distributed software development. *Proceedings of the 13th International Conference on Global Software Engineering*, 102–111. https://doi.org/10.1145/3196369.3196374

82. Manjavacas, A., Vizcaíno, A., Ruiz, F., & Piattini, M. (2020). Global software development governance: Challenges and solutions. *Journal of Software: Evolution and Process*, *32*(10). https://doi.org/10.1002/smr.2266

83. *Marek, K., Wińska, E., & Dąbrowski, W. (2021). The State of Agile Software Development Teams During the Covid-19 Pandemic. In A. Przybyłek, J. Miler, A. Poth, & A. Riel (Eds.), *Lean and Agile Software Development* (Vol. 408, pp. 24–39). Springer International Publishing. https://doi.org/10.1007/978-3-030-67084-9_2

84. Marinho, M., Camara, R., & Sampaio, S. (2021). Toward Unveiling How SAFe Framework Supports Agile in Global Software Development. *IEEE Access*, *9*, 109671–109692. https://doi.org/10.1109/ACCESS.2021.3101963

85. *Marinho, M., Noll, J., Richardson, I., & Beecham, S. (2019). Plan-Driven Approaches Are Alive and Kicking in Agile Global Software Development. *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–11. https://doi.org/10.1109/ESEM.2019.8870168

86. *Matthiesen, S., & Bjørn, P. (2017). When Distribution of Tasks and Skills are Fundamentally Problematic: A Failure Story from Global Software Outsourcing. *Proceedings of the ACM on Human-Computer Interaction*, *1*(CSCW), 1–16. https://doi.org/10.1145/3139336

87. *McCarthy, S., McCarthy, S., O'Raghallaigh, P., OReilly, P., Fitzgerald, C., & Adam, F. (2020). *Building Bridges, Burning Bridges: The Use of Boundary Objects in Agile Distributed ISD Teams*. 10.

88. McCarthy, S., O'Raghallaigh, P., Fitzgerald, C., & Adam, F. (2019). Towards a Framework for Shared Understanding and Shared Commitment in Agile Distributed ISD Project Teams. *ECIS*.

89. *Mishra, A., Sinha, K. K., & Thirumalai, S. (2017). Project Quality: The Achilles Heel of Offshore Technology Projects? *IEEE Transactions on Engineering Management*, *64*(3), 272–286. https://doi.org/10.1109/TEM.2017.2662021

90. *Moe, N. B., Faegri, T. E., Cruzes, D. S., & Faugstad, J. E. (2016). Enabling Knowledge Sharing in Agile Virtual Teams. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 29–33. https://doi.org/10.1109/ICGSE.2016.30

91. *Moe, N. B., Stray, V., & Goplen, M. R. (2020). Studying Onboarding in Distributed Software Teams: A Case Study and Guidelines. *Proceedings of the Evaluation and Assessment in Software Engineering*, 150–159. https://doi.org/10.1145/3383219.3383235

92. Nisyak, A. K., Rizkiyah, K., & Raharjo, T. (2020). Human Related Challenges in Agile Software Development of Government Outsourcing Project. *2020 7th International Conference on Electrical Engineering, Computer Sciences and Informatics (EECSI)*, 222–229. https://doi.org/10.23919/EECSI50503.2020.9251899

93. *Noll, J., Beecham, S., Razzak, A., Richardson, B., Barcomb, A., & Richardson, I. (2017). Motivation and autonomy in global software development. *Lecture Notes in Business Information Processing*, *306*, 19–38. https://doi.org/10.1007/978-3-319-70305-3_2

94. Noll, J., Razzak, A., Richardson, I., & Beecham, S. (2016). Agile Practices for the Global Teaming Model. *2016 IEEE 11th International Conference*

on Global Software Engineering Workshops (ICGSEW), 13–18. https://doi.org/10.1109/ICGSEW.2016.20

95. *Noll, J., Razzak, M. A., & Beecham, S. (2017). Motivation and Autonomy in Global Software Development: An Empirical Study. *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 394–399. https://doi.org/10.1145/3084226.3084277

96. Noor, H., Hayat, B., Amjad, Z., Hanif, M., Tabussum, S., Mansha, R., & Mubasher, K. (2021). Identifying Communication Issues Contributing to the Formation of Chaotic Situation: An AGSD View. *International Journal of Advanced Computer Science and Applications*, *12*(2). https://doi.org/10.14569/IJACSA.2021.0120268

97. *Paasivaara, M. (2017). Adopting SAFe to Scale Agile in a Globally Distributed Organization. *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 36–40. https://doi.org/10.1109/ICGSE.2017.15

98. *Paasivaara, M., & Lassenius, C. (2016). Scaling Scrum in a Large Globally Distributed Organization: A Case Study. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 74–83. https://doi.org/10.1109/ICGSE.2016.34

99. *Pandya, A., Mani, V. S., & Pattanayak, A. (2020). Expanding the responsibility of an offshore team and sustainably increasing business value using SAFe. *Proceedings of the 15th International Conference on Global Software Engineering*, 1–5. https://doi.org/10.1145/3372787.3390441

100. *Pardo Calvache, C. J., Chilito Gomez, P. R., Viveros Meneses, D. E., & Pino Correa, F. J. (2019). Scrum+: A scaled Scrum for the agile global software development project management with multiple models. *Revista*

Facultad de Ingeniería Universidad de Antioquia, *93*, 105–116. https://doi.org/10.17533//udea.redin.20190519

101. *Podari, Z., Arbain, A. F., Ibrahim, N., & Sudarmilah, E. (2021). Risk Mitigation Framework for Agile Global Software Development. *Lecture Notes on Data Engineering and Communications Technologies*, *72*, 1233–1246.

102. *Portela, L. T., & Borrego, G. (2016). Scrumconix: Agile and Documented Method to AGSD. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 195–196. https://doi.org/10.1109/ICGSE.2016.39

103. *Qahtani, A. M. (2020). An Empirical Study of Agile Testing in A Distributed Software Development Project. *Proceedings of the 2020 3rd International Conference on Geoinformatics and Data Analysis*, 110–114. https://doi.org/10.1145/3397056.3397085

104. Rafeek, M. A., Arbain, A. F., & Sudarmilah, E. (2019). Risk mitigation techniques in agile development processes. *International Journal of Supply Chain Management*, *8*(2), 1123–1129.

105. *Raith, F., Richter, I., & Lindermeier, R. (2017). How Project-management-tools are used in Agile Practice: Benefits, Drawbacks and Potentials. *Proceedings of the 21st International Database Engineering & Applications Symposium on - IDEAS 2017*, 30–39. https://doi.org/10.1145/3105831.3105865

106. *Rajpal, M. (2018). Effective distributed pair programming. *Proceedings of the 13th International Conference on Global Software Engineering*, 6–10. https://doi.org/10.1145/3196369.3196388

107. Rashid, N., & Khan, S. U. (2018a). Using agile methods for the development of green and sustainable software: Success factors for GSD

vendors. *Journal of Software: Evolution and Process*, *30*(8), e1927. https://doi.org/10.1002/smr.1927

108. Rashid, N., & Khan, S. U. (2018b). Agile practices for global software development vendors in the development of green and sustainable software. *Journal of Software: Evolution and Process*, *30*(10), e1964. https://doi.org/10.1002/smr.1964

109. Rashid, N., Khan, S. U., Khan, H. U., & Ilyas, M. (2021). Green-Agile Maturity Model: An Evaluation Framework for Global Software Development Vendors. *IEEE Access*, *9*, 71868–71886. https://doi.org/10.1109/ACCESS.2021.3079194

110. *Razzak, M. A. (2016). An Empirical Study on Lean and Agile Methods in Global Software Development. *2016 IEEE 11th International Conference on Global Software Engineering Workshops (ICGSEW)*, 61–64. https://doi.org/10.1109/ICGSEW.2016.22

111. *Razzak, M. A., Richardson, I., Noll, J., Canna, C. N., & Beecham, S. (2018). Scaling agile across the global organization: An early stage industrial SAFe self-assessment. *Proceedings of the 13th International Conference on Global Software Engineering*, 121–130. https://doi.org/10.1145/3196369.3196373

112. *Robinson, P. T., & Entertainment, S. I. (2019). *Communication Network in an Agile Distributed Software Development Team*. 5.

113. *Roman, G., Marczak, S., Dutra, A., & Prikladnicki, R. (2015). On the Agile Transformation in a Large-Complex Globally Distributed Company: Why Boarding this Journey, Steps Taken and Main Foreseen Concerns. *2015 6th Brazilian Workshop on Agile Methods (WBMA)*, 32–39. https://doi.org/10.1109/WBMA.2015.13

114. Roopa, M. S., Mani, V. S., & Stefan, H. (2016). An Approach for Enabling Effective and Systematic Software Reuse: In a Globally Distributed Software Engineering Team That Uses a Lean Development Methodology. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 134–138. https://doi.org/10.1109/ICGSE.2016.14

115. Ruane, E., Smith, R., Bean, D., Tjalve, M., & Ventresque, A. (2020). Developing a conversational agent with a globally distributed team: An experience report. *Proceedings of the 15th International Conference on Global Software Engineering*.

116. *Rzhvskyi, A., Bil, E. A., Witeck, G., Aquere, A. L., Lima, R. M., Granja, J., & Azenha, M. (2020). Communication tools used by distributed teams in a BIM learning project. *International Symposium on Project Approaches in Engineering Education*, *10*, 424–431. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85090846365&partnerID=40&md5=50fd81af43c037178e0ba7a2065b24ac

117. Saikiran, I., & Simon, R. (2019). Agile Software development in distributed team Enhancement Techniques. *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 1147–1151. https://doi.org/10.1109/ICCS45141.2019.9065444

118. *Salameh, A., & Bass, J. (2018). Influential Factors of Aligning Spotify Squads in Mission-Critical and Offshore Projects – A Longitudinal Embedded Case Study. In M. Kuhrmann, K. Schneider, D. Pfahl, S. Amasaki, M. Ciolkowski, R. Hebig, P. Tell, J. Klünder, & S. Küpper (Eds.), *Product-Focused Software Process Improvement* (Vol. 11271, pp. 199–215). Springer International Publishing. https://doi.org/10.1007/978-3-030-03673-7_15

119. *Salikhov, D., Succi, G., & Tormasov, A. (2020). *An Empirical Analysis of Success Factors in the Adoption of the Scaled Agile Framework – First Outcomes from an Empirical Study*. 4.

120. Sarwar, A., Hafeez, Y., Hussain, S., & Yang, S. (2020). Towards Taxonomical-Based Situational Model to Improve the Quality of Agile Distributed Teams. *IEEE Access*, *8*, 6812–6826. https://doi.org/10.1109/ACCESS.2020.2964432

121. *Seckin, I., & Ovatman, T. (2018). An empirical study on scrum application patterns in distributed teams. *Proceedings of the 13th International Conference on Global Software Engineering*, 135–136. https://doi.org/10.1145/3196369.3196381

122. Shafiq, M., Zhang, Q., Akbar, M. A., Kamal, T., Mehmood, F., & Riaz, M. T. (2020). Towards successful global software development. *Proceedings of the Evaluation and Assessment in Software Engineering*, 445–450. https://doi.org/10.1145/3383219.3383283

123. *Shafiq, M., Zhang, Q., Akbar, M. A., Khan, A. A., Hussain, S., Amin, F.-E., Khan, A., & Soofi, A. A. (2018). Effect of Project Management in Requirements Engineering and Requirements Change Management Processes for Global Software Development. *IEEE Access*, *6*, 25747–25763. https://doi.org/10.1109/ACCESS.2018.2834473

124. *Shameem, M., Khan, A. A., Hasan, Md. G., & Akbar, M. A. (2020). Analytic Hierarchy Process Based Prioritisation and Taxonomy of Success Factors for Scaling Agile Methods in Global Software Development. *IET Software*, *14*(4), 389–401. https://doi.org/10.1049/iet-sen.2019.0196

125. *Shameem, M., Kumar, R. R., Kumar, C., Chandra, B., & Khan, A. A. (2018). Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process. *Journal of*

Software: Evolution and Process, *30*(11), e1979. https://doi.org/10.1002/smr.1979

126. Shameem, M., Kumar, R. R., Nadeem, M., & Khan, A. A. (2020). Taxonomical classification of barriers for scaling agile methods in global software development environment using fuzzy analytic hierarchy process. *Applied Soft Computing*, *90*, 106122. https://doi.org/10.1016/j.asoc.2020.106122

127. Shameem, Mohd., Kumar, C., & Chandra, B. (2017). Challenges of management in the operation of virtual software development teams: A systematic literature review. *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 1–8. https://doi.org/10.1109/ICACCS.2017.8014695

128. *Sharma, V. S., & Kaulgud, V. (2016). Agile Workbench: Tying People, Process, and Tools in Distributed Agile Delivery. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 69–73. https://doi.org/10.1109/ICGSE.2016.17

129. *Shrivastava, S. V., & Rathod, U. (2017). A risk management framework for distributed agile projects. *Information and Software Technology*, *85*, 1–15. https://doi.org/10.1016/j.infsof.2016.12.005

130. Sievi-Korte, O., Beecham, S., & Richardson, I. (2019). Challenges and recommended practices for software architecting in global software development. *Information and Software Technology*, *106*, 234–253. https://doi.org/10.1016/j.infsof.2018.10.008

131. *Sievi-Korte, O., Richardson, I., & Beecham, S. (2019). Software architecture design in global software development: An empirical study. *Journal of Systems and Software*, *158*, 110400. https://doi.org/10.1016/j.jss.2019.110400

132. Singh, H., & Bhattacharjee, A. (2019). A Study of Agile Adoption in Project Management with Reference to Issues and Concern Across Various Industries. *Control Systems*, *11*, 13.

133. Sinha, R., Shameem, M., & Kumar, C. (2020). SWOT: Strength, Weaknesses, Opportunities, and Threats for Scaling Agile Methods in Global Software Development. *Proceedings of the 13th Innovations in Software Engineering Conference on Formerly Known as India Software Engineering Conference*, 1–10. https://doi.org/10.1145/3385032.3385037

134. *Šmite, D., Gonzalez-Huerta, J., & Moe, N. B. (2020). "When in Rome, Do as the Romans Do": Cultural Barriers to Being Agile in Distributed Teams. In V. Stray, R. Hoda, M. Paasivaara, & P. Kruchten (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (Vol. 383, pp. 145–161). Springer International Publishing. https://doi.org/10.1007/978-3-030-49392-9_10

135. *Šmite, D., Moe, N. B., & Gonzalez-Huerta, J. (2021). Overcoming cultural barriers to being agile in distributed teams. *Information and Software Technology*, *138*, 106612. https://doi.org/10.1016/j.infsof.2021.106612

136. *Srivastava, P., & Jain, S. (2017). A leadership framework for distributed self-organized scrum teams. *Team Performance Management: An International Journal*, *23*(5/6), 293–314. https://doi.org/10.1108/TPM-06-2016-0033

137. *Stray, V., & Moe, N. B. (2020). Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack. *Journal of Systems and Software*, *170*, 110717. https://doi.org/10.1016/j.jss.2020.110717

138. *Stray, V., Moe, N. B., & Noroozi, M. (2019). Slack Me If You Can! Using Enterprise Social Networking Tools in Virtual Agile Teams. *2019*

ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE), 111–121. https://doi.org/10.1109/ICGSE.2019.00031

139. *Szabo, D. M., & Steghofer, J.-P. (2019). Coping Strategies for Temporal, Geographical and Sociocultural Distances in Agile GSD: A Case Study. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 161–170. https://doi.org/10.1109/ICSE-SEIP.2019.00025

140. *Tanner, M., & Dauane, M. (2017). *The Use of Kanban to Alleviate Collaboration and Communication Challenges of Global Software Development*. 21.

141. Talukder, A. B. M., Senapathi, M., & Buchan, J. (2017). *Coordination in Distributed Agile Software Development: A Systematic Review*.

142. *Usman, M., & Britto, R. (2016). Effort Estimation in Co-located and Globally Distributed Agile Software Development: A Comparative Study. *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 219–224. https://doi.org/10.1109/IWSM-Mensura.2016.042

143. *Vallon, R., Strobl, S., Ras, M., Bernhart, M., & Grechenig, T. (2019). Distributed Kanban with Limited Geographical Distance: Analyzing Lean Principles Pull, Work in Progress and Kaizen: *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering*, 210–217. https://doi.org/10.5220/0007626302100217

144. Vithana, V. N., Asirvatham, D., & Johar, M. G. M. (2017). Investigating the Issues of Using Agile Methods in Offshore Software Development in Sri Lanka. In D. Król, N. T. Nguyen, & K. Shirai (Eds.), *Advanced Topics in Intelligent Information and Database Systems* (Vol. 710, pp. 515–523).

Springer International Publishing. https://doi.org/10.1007/978-3-319-56660-3_44

145. *Vithana, V. N., Asirvatham, D., & Johar, M. G. M. (2018). An Empirical Study on Using Agile Methods in Global Software Development. *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*, 150–156. https://doi.org/10.1109/ICTER.2018.8615505

146. *Wińska, E., & Dąbrowski, W. (2020). Software Development Artifacts in Large Agile Organizations: A Comparison of Scaling Agile Methods. In A. Poniszewska-Marańda, N. Kryvinska, S. Jarząbek, & L. Madeyski (Eds.), *Data-Centric Business and Applications* (Vol. 40, pp. 101–116). Springer International Publishing. https://doi.org/10.1007/978-3-030-34706-2_6

147. *Wong, S. I., & van Gils, S. (2021). Initiated and received task interdependence and distributed team performance: The mediating roles of different forms of role clarity. *AI & SOCIETY*. https://doi.org/10.1007/s00146-021-01241-w

148. Wu, T. (2021). Digital project management: Rapid changes define new working environments. *Journal of Business Strategy*, *ahead-of-print*(ahead-of-print). https://doi.org/10.1108/JBS-03-2021-0047

D.2 Reports Excluded from the Review after Full Text Considered

1. Agrawal, P., Zahaf, K., Sinha, A. K., & Draoui, E. (2019). Agile approach to optimize field development plan with maximum leverage of an EPS phase learnings in offshore Abu Dhabi. *Society of Petroleum Engineers - SPE Reservoir Characterisation and Simulation Conference and Exhibition 2019, RCSC 2019*. http://dx.doi.org/10.2118/196728-ms

2. Akbar, M. A., Shameem, M., Khan, A. A., Nadeem, M., Alsanad, A., & Gumaei, A. (2021). A fuzzy analytical hierarchy process to prioritize the success factors of requirement change management in global software

development. *Journal of Software: Evolution and Process*, *33*(2). https://doi.org/10.1002/smr.2292

3. Duehr, K., Heimicke, J., Breitschuh, J., Spadinger, M., Kopp, D., Haertenstein, L., & Albers, A. (2019). Understanding Distributed Product Engineering: Dealing with Complexity for Situation- and Demand-Oriented Process Design. *Procedia CIRP*, *84*, 136–142. https://doi.org/10.1016/j.procir.2019.04.200

4. Ghane, K. (2017). Quantitative planning and risk management of Agile Software Development. *2017 IEEE Technology & Engineering Management Conference (TEMSCON)*, 109–112. https://doi.org/10.1109/TEMSCON.2017.7998362

5. Gutwin, C., Ochoa, S. F., Vassileva, J., & Inoue, T. (Eds.). (2017). *Collaboration and Technology* (Vol. 10391). Springer International Publishing. https://doi.org/10.1007/978-3-319-63874-4

6. Kabbur, P. K., Mani, V. S., & Schuelein, J. (2020). Prioritizing trust in a globally distributed software engineering team to overcome complexity and make releases a non-event. *Proceedings of the 15th International Conference on Global Software Engineering*, 66–70. https://doi.org/10.1145/3372787.3390434

7. Kramer, W., & Heuvel, J. van den. (2019). The Value of Agile Ways of Working in a Non-Profit Network Organization. *Journal of Creating Value*, *5*(2), 176–189. https://doi.org/10.1177/2394964319860729

8. Lamacchia, D., Chowdhury, K., & Sharif, O. (2020). A novel way of project management to ensure engagement for successful digital transformation. *Proceedings of the Annual Offshore Technology Conference*, *2020-May*. https://www.scopus.com/inward/record.uri?eid=2-s2.0-

85086241490&partnerID=40&md5=19b5326b428bebd9515cea8e05a4efb3

9. Majanoja, A.-M., Linko, L., & Leppänen, V. (2017). Developing offshore outsourcing practices in a global selective outsourcing environment – the IT supplier's viewpoint. *IJISPM - International Journal of Information Systems and Project Management*, *5(1)*, 27–43. https://doi.org/10.12821/ijispm050102

10. Pellicelli, M. (2018). Gaining Flexibility and Innovation through Offshore Outsourcing. *Sustainability*, *10*(5), 1672. https://doi.org/10.3390/su10051672

11. Robe, P., Kaur Kuttal, S., Zhang, Y., & Bellamy, R. (2020). Can Machine Learning Facilitate Remote Pair Programming? Challenges, Insights & Implications. *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–11. https://doi.org/10.1109/VL/HCC50065.2020.9127250

12. Saeedi, K., & Visvizi, A. (2021). Software Development Methodologies, HEIs, and the Digital Economy. *Education Sciences*, *11*(2), 73. https://doi.org/10.3390/educsci11020073

13. Santistevan, D., & Josserand, E. (2019). Meta-Teams: Getting Global Work Done in MNEs. *Journal of Management*, *45*(2), 510–539. https://doi.org/10.1177/0149206318793184

14. Wieland, S., Anke, J., Fichte, A., & Helbig, J. (2017). Adaption von SCRUM für verteilte Teams in Teilzeitarbeit: Eine empirische Analyse studentischer Projekte. *Informatik-Spektrum*, *40*(5), 445–454. https://doi.org/10.1007/s00287-016-1003-4

APPENDIX D. BIBLIOGRAPHIC DETAILS OF REPORTS SYSTEMATICALLY REVIEWED

D.3 Reports Excluded from Review Based on Unavailability of Full Texts

1. Ahmed, Z., Mansor, Z., & Ahmad, K. (2017). An Analysis of Knowledge Management Challenges in Agile Global Software Development. *Journal of Telecommunication, Electronic and Computer Engineering*, *9*(3-4 Special Issue), 63–66.
2. Griffin, L. (2021). Implementing Lean Principles in Scrum to Adapt to Remote Work in a Covid-19 Impacted Software Team. *Lecture Notes in Business Information Processing*, *408*, 177–184. https://doi.org/10.1007/978-3-030-67084-9_11
3. Hossain, S. S., Arafat, Y., Amin, T., & Bhuiyan, T. (2020). Requirements Re-usability in Global Software Development: A Systematic Mapping Study. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12252 LNCS*, 960–974. https://doi.org/10.1007/978-3-030-58811-3_68
4. Modi, S., Abbott, P., & Counsell, S. (2017). Exploring the emergence of collaborative practices in globally distributed agile software development. *AMCIS 2017 - America's Conference on Information Systems: A Tradition of Innovation*, *2017-August*, 1 DUMMY. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85048442429&partnerID=40&md5=3a4c8cf1c7e1e9b566ecacc92a779a96
5. Nundlall, C., & Nagowah, S. D. (2021). Factors Affecting Task Allocation and Coordination in Distributed Agile Software Development. *Advances in Intelligent Systems and Computing*, *1299 AISC*, 817–829. https://doi.org/10.1007/978-981-33-4299-6_66
6. Watson, M. D. M. (2019). From global vision to agile execution: A proposed planning model. *Organizational Cultures*, *19*(1), 13–22. https://doi.org/10.18848/2327-8013/CGP/v19i01/13-22

Bibliography

Abrahamsson, P., Salo, O., & Ronkainen, J. (2002). *Agile Software Development Methods: Review and Analysis*. 113.

Adelakun, O., Garcia, R., Tabaka, T., & Ismail, R. (2017.). *Hybrid Project Management: Agile with Discipline*. 14.

Ahmad, M. O., Lenarduzzi, V., Oivo, M., & Taibi, D. (2018). *Lessons Learned on Communication Channels and Practices in Agile Software Development*. 929–938. https://doi.org/10.15439/2018F72

Akbar, M. A., Sang, J., Nasrullah, Khan, A. A., Mahmood, S., Qadri, S. F., Hu, H., & Xiang, H. (2019). Success factors influencing requirements change management process in global software development. *Journal of Computer Languages*, *51*, 112–130. https://doi.org/10.1016/j.cola.2018.12.005

Allen, T.D., Golden, T.D., & Shockley, K.M. (2015). How effective is telecommuting? Assessing the status of our scientific findings. Psychological Science in the PublicInterest, 16(2), 40–68. https://doi.org/10.1177/1529100615593273

Alzoubi, Y. I., Gill, A. Q., & Al-Ani, A. (2016). Empirical studies of geographically distributed agile development communication challenges: A systematic review. *Information & Management*, *53*(1), 22–37. https://doi.org/10.1016/j.im.2015.08.003

Anderson, K. E. (2016). Getting acquainted with social networks and apps: picking up the Slack in communication and collaboration. Library Hi Tech News, 33(9), 6-9.

Arora, M., Chopra, S., Rakhra, M., Minhas, V., Walia, R., Aggarwal, R., & Kumar, M. (2021). *Agile Umbrella Methodologies and its Global Impact*. *25*(4), 14.

Anwer, F., Aftab, S., Shah, S. S. M., & Waheed, U. (2017). *Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum*. *8*(2), 8.

Ashforth, B. E., Kreiner, G. E., & Fugate, M. (2000). All in a day's work: Boundaries and micro role transitions. Academy of Management Review, 25(3), 472–491. https://doi.org/10.5465/AMR.2000.3363315

Akturk, M. S., & Erhun, F. (1999). An overview of design and operational issues of kanban systems. *International Journal of Production Research*, *37*(17), 3859–3881. https://doi.org/10.1080/002075499189808

Arumugam, C., Vaidayanthan, S., & Karuppuchamy, H. (2018). Global software development: Key Performance measures of team in a SCRUM based agile environment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10963 LNCS*, 672–682. https://doi.org/10.1007/978-3-319-95171-3_53

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods. Vtt Publications.

Anderson, D. J., & Carmichael, A. (2016). *Essential Kanban Condensed*.

Aitken, A., & Ilango, V. (2013). A Comparative Analysis of Traditional Software Engineering and Agile Software Development. *2013 46th Hawaii International Conference on System Sciences*, 4751–4760. https://doi.org/10.1109/hicss.2013.31

Avison, D., & Fitzgerald, G. (2003). Information systems development: methodologies, techniques, and tools. Berkshire, UK: McGraw Hill, 4th ed.

Aveson, D., & Fitzgerald, G. (2006). Methodologies for Developing Information Systems: A Historical Perspective. *The Past and Future of Information Systems: 1976–2006 and Beyond*, 27–38. https://doi.org/10.1007/978-0-387-34732-5_3

Awad, M. A. (2005). A Comparison between Agile and Traditional Software Development Methodologies. *Global Journal of Computer Science and Technology*, 7–42. https://doi.org/10.34257/GJCSTCVOL20IS2PG7

Abbas, N., Gravell, A. M., & Wills, G. B. (2008). Historical Roots of Agile Methods: Where Did "Agile Thinking" Come From? *Lecture Notes in Business Information Processing*, 94–103. https://doi.org/10.1007/978-3-540-68255-4_10.

Alsahli, A., Khan, H., & Alyahya, S. (2017). *Agile Development Overcomes GSD Challenges: A Systematic Literature Review*. 13.

Awar, K. B., Sameem, M. S. I., & Hafeez, Y. (2017). A model for applying Agile practices in Distributed environment: A case of local software industry. *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*, 228–232. https://doi.org/10.1109/C-CODE.2017.7918933

Beck K, Beedle M, Bennekum van A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin R, Mellor S, Schwaber K, Sutherland J, Thomas D (2001) Manifesto for agile software development. Agilemanifesto.org.

Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70–77. https://doi.org/10.1109/2.796139

Beck, K. (2000). Extreme Programming Explained: Embrace Change.

Beck, K. (2004). Extreme Programming Explained: Embrace Change, 2nd ed.

Berry, G. R. (2011). Enhancing Effectiveness on Virtual Teams: Understanding Why Traditional Team Skills Are Insufficient. *Journal of Business Communication*, *48*(2), 186–206. https://doi.org/10.1177/0021943610397270

Boehm, B. (2002). Get ready for agile methods, with care. Computer, 35 (1), 64–69. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=976920

Boehm, B. (2006). A view of 20th and 21st century software engineering. In Proceedings of the 28th international conference on Software engineering, (pp. 12–29). ACM

Bosch-Sijtsema, P. M., & Sivunen, A. (2013). Professional Virtual Worlds Supporting Computer-Mediated Communication, Collaboration, and Learning in

Geographically Distributed Contexts. *IEEE Transactions on Professional Communication*, *56*(2), 160–175. https://doi.org/10.1109/TPC.2012.2237256

Brechner, E., & Waletzky, J. (2015). *Agile project management with Kanban*. Microsoft Press.

Bakshi, S., & Krishna S. (2008). The impact of virtuality on the flexibility of virtual teams in software development projects. In Proceedings of the 14th Americas Conference on Information Systems.

Boden, A., Nett, B., & Wulf, V. (2007). Coordination practices in distributed software development of small enterprises. International Conference on Global Software Engineering (ICGSE), Munich, Germany, IEEE Press: 235–244.

Banijamali, A., Dawadi, R., Ahmad, M. O., Similä, J., Oivo, M., & Liukkunen, K. (2017). Empirical Investigation of Scrumban in Global Software Development. In S. Hammoudi, L. F. Pires, B. Selic, & P. Desfray (Eds.), *Model-Driven Engineering and Software Development* (Vol. 692, pp. 229–248). Springer International Publishing. https://doi.org/10.1007/978-3-319-66302-9_12

Batool, A., Chowdhury, M., & Chowdhury, A. (2019). *A Survey of Key Challenges of Adopting Agile in Global Software Development: A Case Study with Malaysia Perspective*. 7.

Beecham, S., Clear, T., Lal, R., & Noll, J. (2021). Do scaling agile frameworks address global software development risks? An empirical study. *Journal of Systems and Software*, *171*, 110823. https://doi.org/10.1016/j.jss.2020.110823

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2018). Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. *IEEE Transactions on Software Engineering*, *44*(10), 932–950. https://doi.org/10.1109/TSE.2017.2730870

Bjørn, P., Søderberg, A.-M., & Krishna, S. (2019). Translocality in Global Software Development: The Dark Side of Global Agile. *Human–Computer Interaction*, *34*(2), 174–203. https://doi.org/10.1080/07370024.2017.1398092

Brynjolfsson, E., Horton, J., Ozimek, A., Rock, D., Sharma, G., & TuYe, H.-Y. (2020). *COVID-19 and Remote Work: An Early Look at US Data* (No. w27344; p. w27344). National Bureau of Economic Research. https://doi.org/10.3386/w27344

Borrego, G., Moran, A. L., & Palacio, R. (2017). Preliminary Evaluation of a Tag-Based Knowledge Condensation Tool in Agile and Distributed Teams. *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 51–55. https://doi.org/10.1109/ICGSE.2017.14

Camara, R., Alves, A., Monte, I., & Marinho, M. (2020). Agile Global Software Development: A Systematic Literature Review. Proceedings of the 34th Brazilian Symposium on Software Engineering, 31–40. https://doi.org/10.1145/3422392.3422411

Chowdhury, A. F., & Huda, M. N. (2011). Comparison between Adaptive Software Development and Feature Driven Development. *Proceedings of 2011 International Conference on Computer Science and Network Technology*, *1*, 363–367. https://doi.org/10.1109/ICCSNT.2011.6181977

Cobb, C. G. (2015). *The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach* (1st ed.). Wiley. http://3.droppdf.com/files/TIUVU/the-project-manager-s-guide-to-mastering-agile.pdf

Clarke, P., O'Connor, R. V., & Yilmaz, M. (2018). In search of the origins and enduring impact of Agile software development. *Proceedings of the 2018 International Conference on Software and System Process*, 142–146. https://doi.org/10.1145/3202710.3203162.

Corbucci, H., Goldman, A., Katayama, E., Kon, F., Melo, C., & Santos, V. (2011). Genesis and Evolution of the Agile Movement in Brazil -- Perspective from Academia and Industry. *2011 25th Brazilian Symposium on Software Engineering*, 98–107. https://doi.org/10.1109/sbes.2011.26

Casteren, W.V. (2017). The Waterfall Model and the Agile Methodologies: A comparison by project characteristics. 10.13140/RG.2.2.36825.72805.

Cockburn, A. (2002). *Agile Software Development*. Addison-Wesley. https://books.google.co.za/books?id=JxYQ1Zb61zkC

Cockburn, A. (2006). *Agile Software Development: The Cooperative Game*. Pearson Education. https://books.google.co.za/books?id=i39yimbrzh4C

Cole, R., & Scotcher, E. (2015). *Brilliant Agile project management: A practical to using Agile, Scrum and Kanban*.

Chilito, P., Viveros, D., Pardo, C., & Pino, F. J. (2018). Scrum+: An agile guide for the global software development (GSD) multi-model project management. *2018 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 1–6.

Cruz, A. F. da, Godoy, C. P., Santos, L. M. dos, Marinho, L. F., Jardim, M. S., Silva, E. P. da, Pahins, C. A., Fonseca, P., & Giuntini, F. T. (2020). Blueprint Model: An Agile-Oriented Methodology for Tackling Global Software Development Challenges. *Advances in Science, Technology and Engineering Systems Journal*, *5*(6), 353–362. https://doi.org/10.25046/aj050643

Cruzes, D. S., Moe, N. B., & Dyba, T. (2016). Communication between Developers and Testers in Distributed Continuous Agile Testing. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 59–68. https://doi.org/10.1109/ICGSE.2016.27

Dafoulas, G., Maia, C., Ali, A., Augusto, J. C., & Lopez-Cabrera, V. (2017). Understanding Collaboration in Global Software Engineering (GSE) Teams with the Use of Sensors: Introducing a Multi-sensor Setting for Observing Social and Human

Aspects in Project Management. *2017 International Conference on Intelligent Environments (IE)*, 114–121. https://doi.org/10.1109/IE.2017.40

Danait, A. (2005). Agile Offshore Techniques-a Case Study. *Proceedings - AGILE Conference 2005*, *2005*, 214–217. https://doi.org/10.1109/ADC.2005.9

Deshpande, A., Sharp, H., Barroca, L., & Gregory, P. (2016). *Remote Working and Collaboration in Agile Teams*. 17.

Dennerlein, S., Gutounig, R., Gexistinggruber, E., & Schweiger, S. (2016). Web 2.0 Messaging Tools for Knowledge Management? Exploring the Potentials of Slack. In European Conference on Knowledge Management (p. 225). Academic Conferences International Limited.

Duncan, S. (2019). *Understanding Agile Values & Principles: An Examination of the Agile Manifesto*. lulu.com. https://www.infoq.com/minibooks/agile-values-principles/

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221. https://doi.org/10.1016/j.jss.2012.02.033

dos Santos, L. S., L'Erario, A., Pagotto, T., Camilo, J. R. M., Oliveira, F. S., & Fabri, J. A. (2018). A scrum-based process to distributed projects in multidisciplinary teams: A case study. *Proceedings of the 13th International Conference on Global Software Engineering*, 133–134. https://doi.org/10.1145/3196369.3196380

Dudziak, T. (1999). *EXtreme Programming An Overview*.

Ehlers, K. (2011). Agile software development as managed sensemaking, Stellenbosch: University of Stellenbosch.

Ebert, C., Kuhrmann, M., & Prikladnicki, R. (2016). Global Software Engineering: Evolution and Trends. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 144–153. https://doi.org/10.1109/ICGSE.2016.19

BIBLIOGRAPHY

Ebrahim, A. N., Mohammed Shahadat, S. A., & Taha, Z. (2014). *Virtual Teams: A Literature Review*. https://doi.org/10.6084/M9.FIGSHARE.1067906

Esteki, M., Javdani Gandomani, T., & Khosravi Farsani, H. (2020). A risk management framework for distributed scrum using PRINCE2 methodology. *Bulletin of Electrical Engineering and Informatics*, *9*(3), 1299–1310. https://doi.org/10.11591/eei.v9i3.1905

Favare, J. (2002). Managing requirements for business value. *IEEE Software*, *19*(2), 15–17. https://doi.org/10.1109/52.991325

Gemino, A., Horner Reich, B., & Serrador, P. M. (2021). Agile, Traditional, and Hybrid Approaches to Project Success: Is Hybrid a Poor Second Choice? Project Management Journal, 52(2), 161–175. https://doi.org/10.1177/8756972820973082

Guillot, I., Paulmani, G., Kumar, V., & Fraser, S. N. (2017). Case studies of industry-academia research collaborations for software development with Agile. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10391 LNCS*, 196–212. https://doi.org/10.1007/978-3-319-63874-4_15

Gunal, V. (2012). Agile Software Development Approaches and Their History. Enterprise Software Engineering. 1-agile.pdf (uni-bonn.de).

Godoy, C. P., Cruz, A. F., Silva, E. P., Santos, L. M., Zerbini, R. S., & Pahins, C. A. L. (2019). Blueprint Model: A new Approach to Scrum Agile Methodology. *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, 95–99. https://doi.org/10.1109/ICGSE.2019.00014

Gandomani. (2013). OBSTACLES IN MOVING TO AGILE SOFTWARE DEVELOPMENT METHODS; AT A GLANCE. *Journal of Computer Science*, *9*(5), 620–625. https://doi.org/10.3844/jcssp.2013.620.625

Gross, J. M., & McInnis, K. R. (2003). *Kanban Made Simple: Demystifying and Applying Toyota's Legendary Manufacturing Process*. AMACOM. https://books.google.co.za/books?id=fR1WvjSlE9gC

Hardill, I., & Green, A. (2003). Remote working—Altering the spatial contours of work and home in the new economy. *New Technology, Work and Employment*, *18*(3), 212–222. https://doi.org/10.1111/1468-005X.00122

Harrison, N., & Labs, A. (2003). *A Study of Extreme Programming in a Large Company*.

Henry, M. S., Le Roux, D. B., & Parry, D. A. (2021). Working in a post Covid-19 world: Towards a conceptual framework for distributed work. *South African Journal of Business Management*, *52*(1). https://doi.org/10.4102/sajbm.v52i1.2155

Hossain, S. S. (2019). Challenges and Mitigation Strategies in Reusing Requirements in Large-Scale Distributed Agile Software Development: A Survey Result. *Advances in Intelligent Systems and Computing*, *998*, 920–935. https://doi.org/10.1007/978-3-030-22868-2_63

Hossain, E., Bannerman, P. L., & Jeffery, D. R. (2011). Towards an understanding of tailoring scrum in global software development: A multi-case study. *ICSSP '11*.

Highsmith, J. (2001). History: The Agile Manifesto. http://www.agilemanifesto.org/history.html

Highsmith, J., & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer*, *34*(9), 120–127. https://doi.org/10.1109/2.947100

Hoda, R., Salleh, N., Grundy, J., & Tee, H. M. (2017). Systematic literature reviews in agile software development: A tertiary study. *Information and Software Technology*, *85*, 60–70. https://doi.org/10.1016/j.infsof.2017.01.007

Hoda, R., Salleh, N., & Grundy, J. (2018). The Rise and Evolution of Agile Software Development. *IEEE Software*, *35*(5), 58–63. https://doi.org/10.1109/MS.2018.290111318

Hron, M., & Obwegeser, N. (2018). *Scrum in practice: An overview of Scrum adaptations*. 10.

Huang, C.-C., & Kusiak, A. (1996). Overview of Kanban systems. *International Journal of Computer Integrated Manufacturing*, *9*(3), 169–189. https://doi.org/10.1080/095119296131643

Hundermark, P. (2015). Do Better Scrum. Version 3.

Jalali, S., & Wohlin, C. (2012). Global software engineering and agile practices: A systematic review: GLOBAL SOFTWARE ENGINEERING AND AGILE PRACTICES. *Journal of Software: Evolution and Process*, *24*(6), 643–659. https://doi.org/10.1002/smr.561

Jayaratna, N. (2004). Understanding and Evaluating Methodologies: NIMSAD, A Systematic Framework.

Jones, J.M. (2015). In U.S., telecommuting for work climbs to 37%. Gallup. Retrieved from https://news.gallup.com/poll/184649/telecommuting-work-climbs.aspx.

Junior, M. L., & Filho, M. G. (2010). Variations of the kanban system: Literature review and classification. *International Journal of Production Economics*, *125*(1), 13–21. https://doi.org/10.1016/j.ijpe.2010.01.009

Koskela, L., & Howell, G. (2002). The Underlying Theory of Project Management Is Obsolete. In Proceedings of PMI Research Conference. Project Management Institute.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Khan, A. A., Keung, J., Niazi, M., Hussain, S., & Shameem, M. (2019). GSEPIM: A roadmap for software process assessment and improvement in the domain of global software development. *Journal of Software: Evolution and Process*, *31*(1), e1988. https://doi.org/10.1002/smr.1988

Khan, A. A., Shameem, M., Nadeem, M., & Akbar, M. A. (2021). Agile trends in Chinese global software development industry: Fuzzy AHP based conceptual mapping. *Applied Soft Computing*, *102*, 107090. https://doi.org/10.1016/j.asoc.2021.107090

Khan, A. I., Qurashi, R. J., & Khan, U. A. (2012). *A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies*. *8*(4), 11.

Lautert, T., Neto, A. G. S. S., & Kozievitch, N. P. (2019). A Survey on Agile Practices and Challenges of a Global Software Development Team. In P. Meirelles, M. A. Nelson, & C.

Lansmann, S., Schallenmüller, S., & Rigby, M. (2019). *Teams Everywhere – Investigating the Impact of Microsoft Teams on Knowledge Worker*. 6.

Lous, P., Kuhrmann, M., & Tell, P. (2017). Is Scrum Fit for Global Software Engineering? *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 1–10. https://doi.org/10.1109/ICGSE.2017.13

Lous, P., Tell, P., Michelsen, C. B., Dittrich, Y., Kuhrmann, M., & Ebdrup, A. (2018). Virtual by design: How a work environment can support agile distributed software development. *Proceedings of the 13th International Conference on Global Software Engineering*, 102–111. https://doi.org/10.1145/3196369.3196374

Lous, P., Tell, P., Michelsen, C. B., Dittrich, Y., & Ebdrup, A. (2018). From Scrum to Agile: A journey to tackle the challenges of distributed development in an Agile team. *Proceedings of the 2018 International Conference on Software and System Process*, 11–20. https://doi.org/10.1145/3202710.3203149

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, *36*(6), 47–56. https://doi.org/10.1109/mc.2003.1204375.

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software Development Life Cycle AGILE vs Traditional Approaches. *International Conference on Information and Network Technology*, *37*, 162–167.

https://www.researchgate.net/publication/268334807_Software_Development_Life_Cycle_AGILE_vs_Traditional_Approaches

Leybourn, E. (2013). *Directing The Agile Organisation*. IT Governance Publishing; JSTOR. http://www.jstor.org/stable/j.ctt5hh6fh

Leonardi, P. M., Huysman, M., & Steinfield, C. (2013). Enterprise Social Media: Definition, History, and Prospects for the Study of Social Technologies in Organizations. *Journal of Computer-Mediated Communication*, *19*(1), 1–19. https://doi.org/10.1111/jcc4.12029

Lee, S., & Yong, H.-S. (2009). Distributed agile: Project management in a global environment. *Empirical Software Engineering*, *15*, 204–217.

Lindstrom, L., & Jeffries, R. (2004). Extreme Programming and Agile Software Development Methodologies. *Information Systems Management*, *21*(3), 41–52. https://doi.org/10.1201/1078/44432.21.3.20040601/82476.7

Licorish, S.A. and MacDonell, S.G. (2021). How do globally distributed agile teams self-organise? Initial insights from a case study. *arXiv preprint arXiv:2106.10614*.

Marques, A. B., Rodrigues, R., & Conte, T. (2012). Systematic Literature Reviews in Distributed Software Development: A Tertiary Study. *2012 IEEE Seventh International Conference on Global Software Engineering*, 134–143. https://doi.org/10.1109/ICGSE.2012.29

Mahmood, W., Rizvi, S. S., & Munir, S. (2022). *Hindrance to Requirements Engineering During Software Development with Globally Distributed Teams*. 8.

Moe, N. B., Cruzes, D. S., Dyba, T., & Engebretsen, E. (2015). Coaching a Global Agile Virtual Team. *2015 IEEE 10th International Conference on Global Software Engineering*, 33–37. https://doi.org/10.1109/ICGSE.2015.26

Moe, N. B., Faegri, T. E., Cruzes, D. S., & Faugstad, J. E. (2016). Enabling Knowledge Sharing in Agile Virtual Teams. *2016 IEEE 11th International Conference on*

Global Software Engineering (ICGSE), 29–33. https://doi.org/10.1109/ICGSE.2016.30

Misra, S., Kumar, V., Kumar, U., Fantazy, K., & Akhter, M. (2012). Agile software development practices: evolution, principles, and criticisms. *International Journal of Quality & Reliability Management*, *29*(9), 972–980. https://doi.org/10.1108/02656711211272863.

Marek, K., Wińska, E., & Dąbrowski, W. (2021). The State of Agile Software Development Teams During the Covid-19 Pandemic. In A. Przybyłek, J. Miler, A. Poth, & A. Riel (Eds.), *Lean and Agile Software Development* (Vol. 408, pp. 24–39). Springer International Publishing. https://doi.org/10.1007/978-3-030-67084-9_2

Mittleman, D. D., & Briggs, B. O. (1998). Communication technology for teams: electronic collaboration. In E. Sunderstrom & Associates (Eds.), Supporting work team effectiveness: Best practices for fostering high-performance. San Francisco, CA: Jossey-Bass.

Manjavacas, A., Vizcaíno, A., Ruiz, F., & Piattini, M. (2020). Global software development governance: Challenges and solutions. *Journal of Software: Evolution and Process*, *32*(10). https://doi.org/10.1002/smr.2266

Martins, L. L., Gilson, L. L., & Maynard, M. T. (2004). Virtual Teams: What Do We Know and Where Do We Go From Here? *Journal of Management*, *30*(6), 805–835. https://doi.org/10.1016/j.jm.2004.05.002

Masood, Z., Hoda, R., & Blincoe, K. (2020). Real World Scrum A Grounded Theory of Variations in Practice. *IEEE Transactions on Software Engineering*, 1–1. https://doi.org/10.1109/TSE.2020.3025317

McCarthy, john C., & MONK, A. F. (1994). Measuring the quality of computer-mediated communication. *Behaviour & Information Technology*, *13*(5), 311–319. https://doi.org/10.1080/01449299408914611

Mnkandla, E., & Dwolatzky, B. (2004). A Survey of Agile Methodologies. *Transactions of the South African Institute of Electrical Engineers*, *95*, 236–247.

Mohagheghi, P. (2004). *Global Software Development: Issues, Solutions, Challenges*. 27.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72–78. https://doi.org/10.1145/1060710.1060712

Nilles, J. M. (1988). Traffic reduction by telecommuting: A status review and selected bibliography. *Transportation Research Part A: General*, *22*(4), 301–317. https://doi.org/10.1016/0191-2607(88)90008-8

Noll, J., Razzak, M. A., & Beecham, S. (2017). Motivation and Autonomy in Global Software Development: An Empirical Study. *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, 394–399. https://doi.org/10.1145/3084226.3084277

Noll, J., & Beecham, S. (2019). How Agile Is Hybrid Agile? An Analysis of the HELENA Data. In X. Franch, T. Männistö, & S. Martínez-Fernández (Eds.), *Product-Focused Software Process Improvement* (Vol. 11915, pp. 341–349). Springer International Publishing. https://doi.org/10.1007/978-3-030-35333-9_25

Noor, H., Hayat, B., Amjad, Z., Hanif, M., Tabussum, S., Mansha, R., & Mubasher, K. (2021). Identifying Communication Issues Contributing to the Formation of Chaotic Situation: An AGSD View. *International Journal of Advanced Computer Science and Applications*, *12*(2). https://doi.org/10.14569/IJACSA.2021.0120268

Oomen, S. (2017). *How can scrum be successful? Competences of the scrum product owner*. 13.

Ozimek, A. (2020). The Future of Remote Work. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3638597

O'Neill, T. A., Hambley, L. A., & Chatellier, G. S. (2014). Cyberslacking, engagement, and personality in distributed work environments. *Computers in Human Behavior*, *40*, 152–160. https://doi.org/10.1016/j.chb.2014.08.005

Paasivaara, M., & Lassenius, C. (2016). Scaling Scrum in a Large Globally Distributed Organization: A Case Study. *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 74–83. https://doi.org/10.1109/ICGSE.2016.34

Parsons, D., Thorn, R., Inkila, M., & MacCallum, K. (2018). Using Trello to Support Agile and Lean Learning with Scrum and Kanban in Teacher Professional Development. *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 720–724. https://doi.org/10.1109/TALE.2018.8615399

Paul, R., & Behjat, L. (2019). *Using principles of scrum project management in an integrated design project*. 14.

Perry, B. 2008. Virtual teams now a reality: Two out of three companies say they will rely more on virtual teams in the future. http://www.pr.com/press-release/103409.

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). *Systematic Mapping Studies in Software Engineering*. 12th International Conference on Evaluation and Assessment in Software Engineering (EASE). https://doi.org/10.14236/ewic/EASE2008.8

Pardo Calvache, C. J., Chilito Gomez, P. R., Viveros Meneses, D. E., & Pino Correa, F. J. (2019). Scrum+: A scaled Scrum for the agile global software development project management with multiple models. *Revista Facultad de Ingeniería Universidad de Antioquia*, *93*, 105–116. https://doi.org/10.17533//udea.redin.20190519

Popli, R., & Chauhan, N. (2011). *SCRUM: AN AGILE FRAMEWORK*. 4.

Qureshi, R. J. M., & Hussain, S. A. (2008). An adaptive software development process model. *Advances in Engineering Software*, *39*(8), 654–658. https://doi.org/10.1016/j.advengsoft.2007.08.001

Rizvi, B., Bagheri, E., & Gasevic, D. (2015). A systematic review of distributed Agile software engineering: Distributed agile software engineering. *Journal of Software: Evolution and Process*, *27*(10), 723–762. https://doi.org/10.1002/smr.1718

Riemer, K., Steinfield, C., & Vogel, D. (2009). eCollaboration: On the nature and emergence of communication and collaboration technologies. *Electronic Markets*, *19*(4), 181–188. https://doi.org/10.1007/s12525-009-0023-1

Rajib, M. (2014). *Introduction to Software Engineering (Version 2)*.

Rashid, N., & Khan, S. U. (2018). Agile practices for global software development vendors in the development of green and sustainable software. *Journal of Software: Evolution and Process*, *30*(10), e1964. https://doi.org/10.1002/smr.1964

Raghuram, S., Hill, N. S., Gibbs, J. L., & Maruping, L. M. (2019). Virtual Work: Bridging Research Clusters. *Academy of Management Annals*, *13*(1), 308–341. https://doi.org/10.5465/annals.2017.0020

Rajpal, M. (2018). Effective distributed pair programming. *Proceedings of the 13th International Conference on Global Software Engineering*, 6–10. https://doi.org/10.1145/3196369.3196388

Raith, F., Richter, I., & Lindermeier, R. (2017). How Project-management-tools are used in Agile Practice: Benefits, Drawbacks and Potentials. *Proceedings of the 21st International Database Engineering & Applications Symposium on - IDEAS 2017*, 30–39. https://doi.org/10.1145/3105831.3105865

Raza, B., MacDonell, S., & Clear, T. (2013). Research in Global Software Engineering: A Systematic Snapshot. *Communications in Computer and Information Science*, *417*, 126–140. https://doi.org/10.1007/978-3-642-54092-9_9

Rajagopalan, S., & Mathew, S. K. (2016). *Choice of Agile Methodologies in Software Development: A Vendor Perspective*. *25*(1), 17.

Rzhvskyi, A., Bil, E. A., Witeck, G., Aquere, A. L., Lima, R. M., Granja, J., & Azenha, M. (2020). Communication tools used by distributed teams in a BIM learning project.

International Symposium on Project Approaches in Engineering Education, *10*, 424–431. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85090846365&partnerID=40&md5=50fd81af43c037178e0ba7a2065b24ac

Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8–13. https://doi.org/10.1145/1764810.1764814

Royce, W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques. In Proceedings of IEEE WESCON, (pp. 1–9).

Roman, G., Marczak, S., Dutra, A., & Prikladnicki, R. (2017). On the Agile Transformation in a Large-Complex Globally Distributed Company: Why Boarding this Journey, Steps Taken and Main Foreseen Concerns. *Proceedings - 6th Brazilian Workshop on Agile Methods, WBMA 2015*, 32–39. https://doi.org/10.1109/WBMA.2015.13

Rockmann, K. W., & Pratt, M. G. (2015). Contagious Offsite Work and the Lonely Office: The Unintended Consequences of Distributed Work. *Academy of Management Discoveries*, *1*(2), 150–164. https://doi.org/10.5465/amd.2014.0016

Roopa, M. S., Kumar, R., & Mani, V. S. (2018). Transitioning from plan-driven to lean in a global software engineering organization: A practice-centric view. *Proceedings - International Conference on Software Engineering*, 1–5. https://doi.org/10.1145/3196369.3196395

Rudnicka, A., Newbold, J. W., Cook, D., Cecchinato, M. E., & Cox, A. L. (2020). *Eworklife: Developing effective strategies for remote working during the COVID-19 pandemic*. 13.

Schwaber, K., & Sutherland, J. (2017). The Scrum Guide.

Stray, V., & Moe, N. B. (2020). Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack. *Journal of Systems and Software*, *170*, 110717. https://doi.org/10.1016/j.jss.2020.110717

Stare, A. (2014). Agile Project Management in Product Development Projects. *Procedia - Social and Behavioral Sciences*, *119*, 295–304. https://doi.org/10.1016/j.sbspro.2014.03.034

Sriram, R., & Mathew, S. K. (2012). Global software development using agile methodologies: A review of literature. *2012 IEEE International Conference on Management of Innovation & Technology (ICMIT)*, 389–393. https://doi.org/10.1109/ICMIT.2012.6225837

Sommerville, I. (2010). *Software Engineering (9th Edition)* (9th ed.). Pearson.

Sutherland, J., Schoonheim, G., Rustenburg, E., & Rijk, M. (2008). Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams. *Agile 2008 Conference*, 339–344. https://doi.org/10.1109/Agile.2008.92

Sutherland, J., & Schwaber,K. (2007). The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process.

Sutherland, J. (2004). Agile development: Lesson learned from the first scrum. *Cutter Agile Project Management Advisory Service: Executive Update, 5(20), 1-4.*

Sutherland, J. (2010). Jeff Sutherland's Scrum Handbook.

Sewell, G., & Taskin, L. (2015). Out of sight, out of mind in a new world of work? Autonomy, control, and spatiotemporal scaling in telework. Organization Studies, 36(11), 1507–1529. https://doi.org/10.1177/0170840615593587

Seckin, I., & Ovatman, T. (2018). An empirical study on scrum application patterns in distributed teams. *Proceedings of the 13th International Conference on Global Software Engineering*, 135–136. https://doi.org/10.1145/3196369.3196381

Shameem, M., Kumar, R. R., Kumar, C., Chandra, B., & Khan, A. A. (2018). Prioritizing challenges of agile process in distributed software development environment using analytic hierarchy process. *Journal of Software: Evolution and Process*, *30*(11), e1979. https://doi.org/10.1002/smr.1979

Shrivastava, S. V., & Date, H. (2010). *Distributed Agile Software Development: A Review*. 8.

Schweitzer, L., & Duxbury, L. (2010). Conceptualizing and measuring the virtuality of teams. *Information Systems Journal*, *20*(3), 267–295. https://doi.org/10.1111/j.1365-2575.2009.00326.x

Sinha, R., Shameem, M., & Kumar, C. (2020). SWOT: Strength, Weaknesses, Opportunities, and Threats for Scaling Agile Methods in Global Software Development. *Proceedings of the 13th Innovations in Software Engineering Conference on Formerly Known as India Software Engineering Conference*, 1–10. https://doi.org/10.1145/3385032.3385037

Szabo, D. M., & Steghofer, J.-P. (2019). Coping Strategies for Temporal, Geographical and Sociocultural Distances in Agile GSD: A Case Study. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 161–170. https://doi.org/10.1109/ICSE-SEIP.2019.00025

Šmite, D., Gonzalez-Huerta, J., & Moe, N. B. (2020). "When in Rome, Do as the Romans Do": Cultural Barriers to Being Agile in Distributed Teams. In V. Stray, R. Hoda, M. Paasivaara, & P. Kruchten (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (Vol. 383, pp. 145–161). Springer International Publishing. https://doi.org/10.1007/978-3-030-49392-9_10

Šmite, D., Wohlin, C., Gorschek, T., & Feldt, R. (2010). Empirical Evidence in Global Software Engineering: A Systematic Review. *Empirical Softw. Engg.*, *15*(1), 91–118. https://doi.org/10.1007/s10664-009-9123-y

Srivastava, P., & Jain, S. (2017). A leadership framework for distributed self-organized scrum teams. *Team Performance Management: An International Journal*, *23*(5/6), 293–314. https://doi.org/10.1108/TPM-06-2016-0033

Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*, *17*(4/2013), 64–76. https://doi.org/10.12948/issn14531305/17.4.2013.06

Svalestuen, F., & Knotten, V., Laedre, O., Drevland, F., Lohne, J., & Lohne, J. (2017). *Using building information model (bim) devices to improve information flow and collaboration on construction sites*. 16.

Thomas, G. F. (2007). How can we make our research more relevant? Bridging the gap between workplace changes and business communication research. Journal of Business Communication, 44, 283-296.

Takeuchi,H., & Nonka,I. (1986). The new product development game. Journal of Product Innovation Management, 3(3), 205-206.

US Office of Personal Management. (2013). *2013 status of telework in the federal government: Report to the Congress.* US Office of Personal Management, DC: Washington.

Vallon, R., da Silva Estácio, B. J., Prikladnicki, R., & Grechenig, T. (2018). Systematic literature review on agile practices in global software development. *Information and Software Technology*, *96*, 161–180. https://doi.org/10.1016/j.infsof.2017.12.004

Van Osch, W., Steinfield, C. W., & Balogh, B. A. (2015). Enterprise Social Media: Challenges and Opportunities for Organizational Communication and Collaboration. *2015 48th Hawaii International Conference on System Sciences*, 763–772. https://doi.org/10.1109/HICSS.2015.97

Vroman, K., & Kovachich, J. (2002). Computer-mediated interdisciplinary teams: Theory and reality. Journal of Interprofessional Care, 16, 159-170.

Vithana, V. N., Asirvatham, D., & Johar, M. G. M. (2018). An Empirical Study on Using Agile Methods in Global Software Development. *2018 18th International*

Conference on Advances in ICT for Emerging Regions (ICTer), 150–156. https://doi.org/10.1109/ICTER.2018.8615505

Volkan, G. (2012). *Agile Software Development Approaches and Their History*.

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2012) Systematic literature reviews in global software development: a tertiary study. 16th International Conference on Evaluation & Assessment in Software Engineering. https://doi.org/10.1049/IC.2012.0001

Williams, L. (2003). The xp programmer: The few-minutes programmer. *IEEE Software*, *20*(3), 16–20. https://doi.org/10.1109/MS.2003.1196315

Wińska, E., & Dąbrowski, W. (2020). Software Development Artifacts in Large Agile Organizations: A Comparison of Scaling Agile Methods. In A. Poniszewska-Marańda, N. Kryvinska, S. Jarząbek, & L. Madeyski (Eds.), *Data-Centric Business and Applications* (Vol. 40, pp. 101–116). Springer International Publishing. https://doi.org/10.1007/978-3-030-34706-2_6

Wong, Y.-M. S. (2018). Virtual sensemaking and self-presentation on Slack: *Exploring the effects of Enterprise Social Network (ESN) on workplace culture and socialisation*. 73.

Webster, J., & Watson, R.T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. MIS Quarterly, 26(2), xiii – xxiii.

Waizenegger, L., McKenna, B., Cai, W., & Bendz, T. (2020). An affordance perspective of team collaboration and enforced working from home during COVID-19. *European Journal of Information Systems*, *29*(4), 429–442. https://doi.org/10.1080/0960085X.2020.1800417

Weber, M. S., & Shi, W. (2016). Enterprise Social Media. In C. R. Scott, J. R. Barker, T. Kuhn, J. Keyton, P. K. Turner, & L. K. Lewis (Eds.), *The International Encyclopedia of Organizational Communication* (1st ed., pp. 1–9). Wiley. https://doi.org/10.1002/9781118955567.wbieoc072

Xu, B. (2009). Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects. *2009 International Conference on Management and Service Science*, 1–4. https://doi.org/10.1109/ICMSS.2009.5302042

Yasvi, M. A. (2019). *Review On Extreme Programming-XP*. 8.

Young, C., & Terashima, H. (2008). How Did We Adapt Agile Processes to Our Distributed Development? *Agile 2008 Conference*, 304–309. https://doi.org/10.1109/Agile.2008.7

Zhang, P., Aikman, S., & Sun, H. (2008). Two types of attitudes in ICT acceptance and use. International Journal of Human Interaction, 24(7), 628-648. doi: 10.1080/10447310802335482.

Zeuge, A., Weigel, A., Niehaves, B., Oschinsky, F., & Schlechtinger, M. (2020). *Leading Virtual Teams – A Literature Review*. 10.