Article

Not peer-reviewed version

Systematic Literature Review Report on the Challenges of Agile Scrum

Melaku Girma Lemma * , Nuno Garcia , Mesfin Kifle

Posted Date: 2 June 2025

doi: 10.20944/preprints202506.0080.v1

Keywords: agile scrum; systematic review; issues of agile scrum; successes of scrum; challenges of agile scrum; agile scrum SLR

 doi:10.20944/preprints202506.0080.v1

Article

Systematic Literature Review Report on the Challenges of Agile Scrum

Melaku Girma [1,*], Nuno Garcia [2] and Mesfin Kifle [3]

[1] Addis Ababa University; melaku.girma@aau.edu.et
[2] University of Lisbon; nmgarcia@fc.ul.pt
[3] Addis Ababa University; mesfin.kifle@aau.edu.et
* Correspondence: melaku.girma@aau.edu.et; Tel.: (+251 977 32 73 82)

Abstract: Agile in general and SCRUM Agile in particular have widely been employed in the software development arena ranging from small to medium to large and multinational projects. There are empirical studies, experience reports, and case studies that have reported the successes, issues and challenges of using the methodology in large-scale software development projects context. This Systematic Literature review employs the PRISMA methodology for searching case studies, experience reports and empirical research related to successes, issues, and challenges faced by software companies that fully harnessed Agile Scrum method for large-scale software development projects. A search protocol and context were prepared to guide the search task. Several papers published over more than a decade from multiple scientific databases, including IEEE Xplore and SCOPUS were analyzed. An iterative coding of themes had been employed to identify and categorize the key themes in the papers. Accordingly, eight main themes and 46 subthemes had been identified. These are Dependency issues, Agile Difficult to Implement, Multi-team Environment Challenges, Challenges of Requirement Engineering, Knowledge Issues, Resistance to Change, Organizational Structure and Boundaries, and Quality Assurance Challenges. Agile Difficult to Implement, Multi-team Environment Challenges and Challenges of Requirement Engineering are the most mentioned themes in order.

Keywords: Agile Scrum; Systematic Review; Issues of Agile Scrum; Successes of Scrum; Challenges of Agile Scrum; Agile Scrum SLR

1. Introduction

Agile is a set of methodologies that share common software development philosophical foundations and practices. However, this paper focuses on one of the highly harnessed Agile methodologies called Scrum. Scrum framework composed of roles, ceremonies, and artifacts [1]. The three distinct roles in the Scrum process are the Product Owner, the Team and the Scrum master [2]. Daily Scrum Meeting, the Daily Scrum of Scrums Meetings, the Sprint Review Meetings and the Sprint Planning Meetings constitute Agile Scrum ceremonies. The Agile Scrum process also employs three artifacts, known as Product Backlog, the Sprint Backlog, and the Burndown Chart.

Agile in general and Agile SCRUM in particular method has intensively been harnessed to meet the ever-increasing market need for short development cycles and quick lead time to release software products [3].

2. Background

Large Software companies such as Google, Apple, Ericson, and Amazon are venturing into Agile SCRUM as there has been an increasing need to develop competencies in continuous software engineering. The multinational and big companies realize the competitive advantages that agility can provide [4]. Despite the high promises of delivering quality software with short lead time in an iterative process, the existing agile SCRUM framework should be tailored to the needs of the software

 doi:10.20944/preprints202506.0080.v1

development companies to meet their large-scale, multi and diverse teams development environment [5]. In this paper, we adopted the definition of large-scale software development as "software development organizations with 50 or more people or at least six teams" [6].

This systematic literature review is envisaged to investigate the body of knowledge on successes, issues and challenges faced by large-scale software development companies that fully harnessed Agile SCRUM for large software development projects. A systematic literature review (often referred to as a systematic review) is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest. It is carried out to summarize the existing evidence concerning a certain topic of interest, to identify any gaps in current research in order to suggest areas for further investigation, and/or, to provide a framework/background in order to appropriately position new research activities [7].

A search protocol and context were prepared to guide the searching task. Several papers published over more than a decade from scientific databases from IEEE Xplore and SCOPUS were analyzed. An iterative coding of themes had been employed to identify and categorize the key themes in the papers. The main contributions of this systematic review are identification and categorizations of substantive themes derived from journal articles identified through the search processes. These themes are contributions to the existing body of knowledge in Agile software development arena and useful references to the industry in its attempt to harness Agile SCRUM as the premier software development method.

Similar Systematic Review had been carried out by [6]. Our review adds more context and themes to their findings. However, a detailed comparison of our work against this review has not been made.

3. Research Methods

The systematic review is carried out by employing the methodological framework of the "Preferred Reporting Items for Systematic Reviews and Meta-Analyses" (PRISMA), harnessing specific inclusion and exclusion protocols and clearly articulated search phrases to search and collect articles highly relevant to the research topic. Then, duplicates, additional unrelated or incomplete articles, and papers that slept through a thorough screening process are removed from the results [8]. The review also follows the guidelines for performing Systematic Literature Reviews in Software Engineering [7].

3.1. Research Questions

This paper presents the results of a systematic literature review on the topic of challenges and scaling practices in using agile Scrum for Large-scale software development projects. Industry experiences reports, case studies, and empirical research works in the field of software engineering have systematically been reviewed. The review processes have been performed in view of the following research question.

RQ 1. What are the specific problems that software companies face when trying to implement Agile SCRUM method, in particular related to the implementation of large projects?

This paper presents the findings of the quest for challenges of Agile SCRUM for large software projects.

3.2. Search Strategy

The search approach in this paper harnesses PRISMA technique to find and analyze case studies, experience reports, empirical research published between January 2011 and October 2023. We employed key terms search approach to identify these papers in IEEE Xplore digital libraries and SCOPUS database. The advanced search facilities of the library and database were used based on the selected search keywords and phrases. Then all articles returned by these systems were analyzed for

 doi:10.20944/preprints202506.0080.v1

selecting relevant papers while eliminating irrelevant ones. The iteratively improved search facets, phrases and keywords in the research are exhibited in Table 1.

Table 1. Search Facets and Keywords.

Facet	Keywords
Agile Method	Scrum OR Agile AND Scrum
Development Context	Large OR "Large Scale" OR "Large team" OR "Large Project" OR "Large Software" OR "Large Software Development"
Scrum Issues	*challenges* OR practices OR "*success factors" OR* application OR use

3.3. Inclusion Criteria

We defined four facets to guide our inclusion/exclusion decisions: Agile software development, large-scale, Context and source type. In addition, the year and language of publication as well as number of citations were used to discriminate irrelevant papers while selecting relevant ones. Table 2 lists the facets and gives examples on matching topics and non-relevant topics. A primary study that fully meets the facets was included in the search result.

Table 2. Inclusion Criteria.

Facet	Relevant topics	Example of non-relevant topics
Agile Software Development	The Company is a software organization that employs Scrum Methodology	Agile Scrum manufacturing; Scrum in management boards Comparison
Large Scale	Scrum/Agile Scrum applied in large scale software development projects	Scaling up from small; a single agile team in a large setting
Context	Challenges, Practices, Success factors, Application and Use	comparison of methodologies; comparison of before and after
Source Type	Journal	Books, Book Series and Conference Proceedings
Year of Publication	2011-2023	Any publication out of this range
Publication Language	English	Publication language except English
Number of Citation	>=3	<3

The actual title, abstract, and keyword search used on the SCOPUS and IEEExplore is exhibited in Table 3.

Table 3. Title Abstract Keyword Search.

TITLE-ABS-KEY ((scrum OR agile AND scrum) AND (large OR "Large Scale" OR "Large team" OR "Large Project" OR "Large Software" OR "Large Software Development") AND (challenges OR practices OR "success factors" OR application OR use)) AND PUBYEAR > 2010 AND

 doi:10.20944/preprints202506.0080.v1

PUBYEAR < 2024 AND (LIMIT-TO (PUBSTAGE , "final")) AND (EXCLUDE (PREFNAMEAUID , "Undefined")) AND (LIMIT-TO (LANGUAGE , "English")) AND (LIMIT-TO (DOCTYPE , "ar"))

3.4. Research Processes

The research process embraces four main phases as depicted in Figure. 1. The primary studies selection was carried out in two phases, phase one was using keyword-based database searches to identify potentially relevant sources, and then the researchers manually extracted the search results. Data extraction was performed by qualitative coding of the selected primary studies. Finally, the results were elicited by aggregating and analyzing the coding of the primary documents.

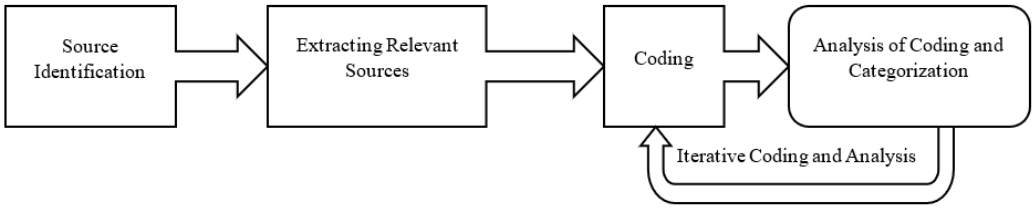

Figure 1. Outline of Research Process.

We developed a context code to categorize the identified papers for review into business area, organization size, and research processes as exhibited in Table 4. The business area delineates the unique area it operates in. The size of the organization was identified provided it was clearly mentioned in the respective empirical studies. The research process employed by each of the papers selected for review has been identified. The context coding result is lucidly exhibited in Appendix B and Appendix C.

Table 4. The Context Code used to Categorize papers Identified for the Review.

Context Code	Explanation
Business area	The business area in which the organization operates
Organization Size	Identifying the size of the organization whenever possible
Research Process	The employed research process in each primary study

4. Analysis

As explained in the foregoing discussion, the search protocols were applied to searching in SCOPUS and IEEExplore databases. A total of 183 papers were returned as depicted in Table 5. After the inclusion and exclusion facets were applied, twenty-six papers were selected for full review. After full review of these papers, only eighteen of them were considered for the study. Eight papers were excluded because they didn't qualify the research topic and overall context of the review. The inclusion and exclusion processes are exhibited in the PRISMA format, as depicted in Figure 2. These eighteen papers selected for review year of publication are lucidly displayed in Table 6. The list of papers reviewed are depicted in Appendix D.

 doi:10.20944/preprints202506.0080.v1

Table 1. Aggregate Search Results.

Database	URL	Number of Matches
IEEExplore	http://ieeexplore.ieee.org	5
Scopus	http://www.scopus.com/home.url	178

Table 2. Papers for review year of publication.

Year	Frequency
2018	4
2017	2
2016	2
2015	3
2014	2
2013	2
2012	2
2011	1

 doi:10.20944/preprints202506.0080.v1

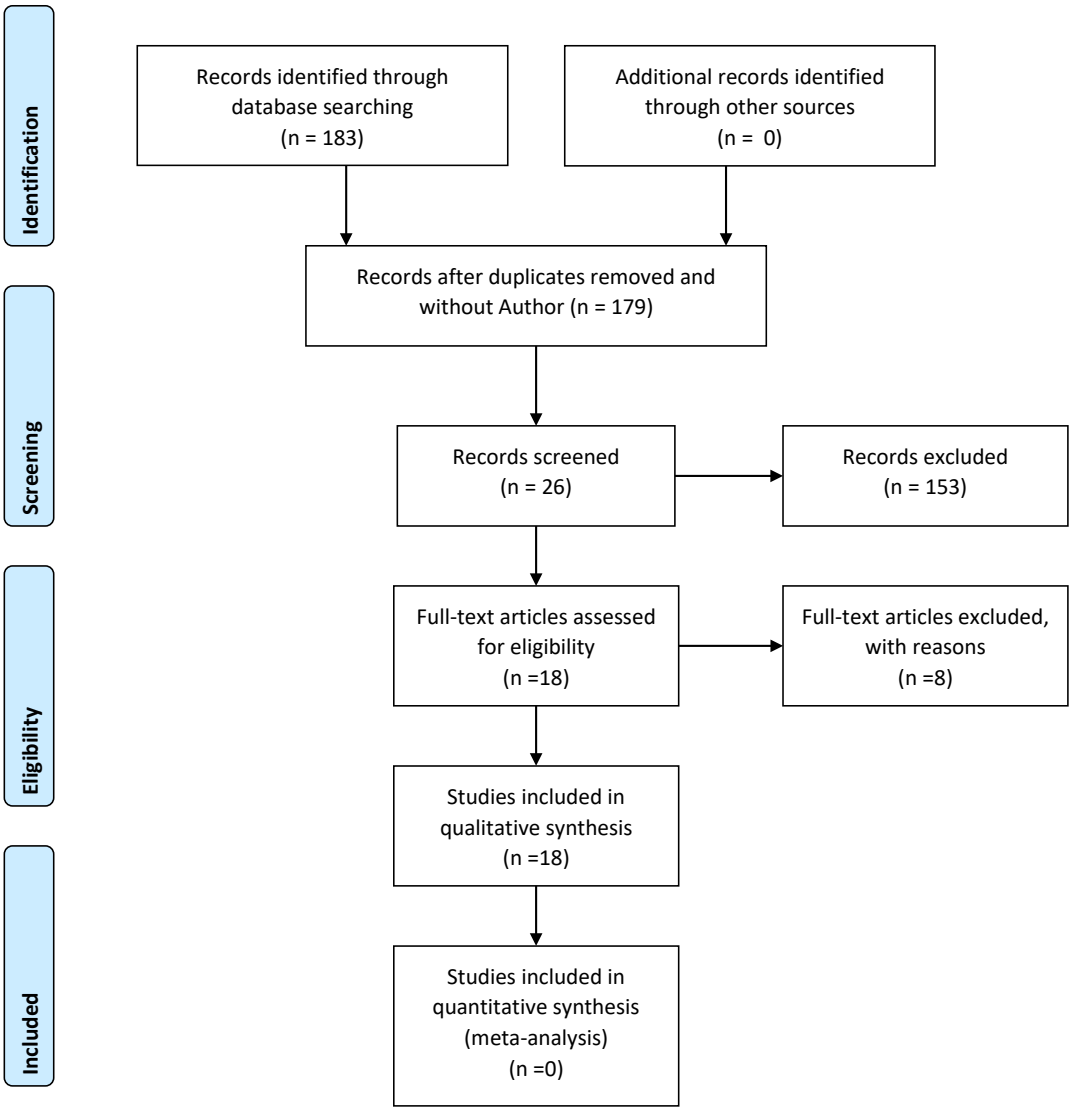

Figure 2. The PRISMA Model of the Papers Selection Processes.

The eight substantive themes identified through iterative coding are described in the subsequent sections and subsections. Appendix A exhibits these themes and subthemes.

4.1. Agile Scrum Challenges Themes and Subthemes

Eight main themes were identified through iterative coding of agile Scrum large-scale implementation issues and challenges. Each of them contains many subthemes. The following sections describe the themes and subthemes identified through iterative coding.

4.1.1. Challenges of using Agile Scrum/Results

Agile methods were initially aimed at small, collocated teams. Due to the success stories of small-scale agile projects, it has been widely used to large-scale and mission/safety-critical software development projects involving multiple teams composed of across different geographic locations [9]. Despite the increasing demands of using agile Scrum for large-scale software developments, many challenges have been reported. This section is meant to provide answer (s) for research questions, RQ1: What are the specific problems that software companies face when trying to implement Agile Scrum method, in particular related to the implementation of large projects? We organized the identified 35 challenges, each mentioned by several sources, into nine categories. The categories are summarized in Appendix A.

 doi:10.20944/preprints202506.0080.v1

4.1.1.1. Dependency Issues

General Lack of Task Dependency Awareness. In this paper, we are using dependency and interdependency interchangeably. In large-scale software development many tasks are interdependent. Dependency exists between people, groups, tasks, and artefacts, including the software components under construction. It is paramount important that stakeholders in the software development recognize this fact. Lack of dependency awareness leads to ineffective coordination in Agile Scrum Development [P01]. For example, in large agile development, there are hundreds of teams working on the same project, which might have interdependent features. Lack of awareness of dependencies among the agile inter-teams' development resulted ineffective coordination [P02,P09]. Stakeholders in the software development should make sure that there is no interdependency between two different features, otherwise undesirable coupling arises [P07,P15].

Dependency happens when the progress of one action relies upon the timely output of a previous action, or the presence of specific thing [P08]. In the application of Agile Scrum for large-scale software development, knowledge, process, and resource dependencies along with their defining characteristics have been identified [P08].

It is reported that agile teams are self-organizing and choose what to develop in a sprint by themselves without external interferences, at least in agile principle. Unless features selected for development by one team is not done in view of other teams' features, features dependency is going to occur, which impedes the development progress[P15].

Expertise Dependency. Agile relies on people and self-organizing teams that are empowered to choose the software features they want to develop in time-boxed fashion, called sprint, by themselves than processes. To this end, agile process lends itself to expertise dependency, which refers to technical or task information is known only by a particular person or group and this affects, or has the potential to affect, project progress [P08].

In Software development there are three expertise. These are technical expertise (knowledge about a specialized technical area), design expertise (knowledge about software design principles and architecture), and domain expertise (knowledge about the application domain area and client operations) [P08]. As agile development heavily emphasizes in the expertise of people, this may lead to expertise dependency, which adversely affects large-scale software projects' success if the expertise is not readily available in a timely fashion, particularly in a time-boxed development [P09,P11,P12].

Business Process and Historical Dependency. Business process dependency is a situation wherein an existing business process causes tasks to be carried out in a certain order, and this affects, or has the potential to affect, project progress, whereas Historical dependencies are defined as the need to mine organizational memory or old code versions for previous decisions [P08]. In large bureaucratic organizations wherein, there are hierarchical and top-down relationships among different entities, both dependencies said to have occurred in agile scrum projects [P08,P11].

Agile implementation in large organizations that has codified and written bureaucratic processes, rules and regulations, should recognize these dependencies and work towards mitigating their impact on the software projects.

Entity Dependency. In software development, entities refer to as physical things such as people, servers, data, and documents. Entity dependency is as a situation wherein a resource (person, place, or thing) is not available, and this affects, or has the potential to affect, project progress [P08]. For example, the entity dependency between development teams and User Experience (UX) design team, where the development teams couldn't proceed with the development as they were waiting input from the UX team [P18].

Technical Dependency. This occurs when technical aspect of development affects progress, such as when one software component must interact with another software component, and its presence or absence affects or has the potential to affect, project progress [P08]. This is further elaborated in [P14], where agile projects that are integrated with software product line should acknowledge the complex technical infrastructure, data security issues and system portfolio. Each of these items are interdependent and don't exist in isolation. There are technical dependencies between

 doi:10.20944/preprints202506.0080.v1

the back-end and front-end developments [P15]. The front-end agile teams should not be developing applications without creating a big-picture of the technicalities of the back-end development teams. In addition, the research in [P18] identified the technical dependencies between agile development and UX design teams. The agile feature development teams were forced to re-work because of the late submissions of the UX design team. Hence, technical dependency is one of the challenges in successfully employing agile in large-scale software development projects.

Social Loafing. A concept taken from social psychology that defines a phenomenon of a person exerting less effort to achieve a goal when he or she works in a group than when working alone [10]. As agile framework recognizes and empowers teams working together to achieve a goal, this phenomena had been reported in two case studies [P04,P10].

4.1.1.2. Agile Difficult to Implement

Old Software Culture to move to Agile and Lean. Change is inevitably happening in today fast-pace growing world. However, it is not easy to give-up or overthrow the old culture and swiftly shift to the new culture. Despite the many success case studies and experience reports, organizations still found it difficult to come by to fully shift their culture into Agile and Lean development. Particularly, the upper management requirements of documentation norms is a challenge in instilling agile culture in the minds of individuals in the organization[P15,P17]. In Agile projects the old software culture assumes initial estimates are fixed and committed to [P06].

Paasivaara and Ebert in their case study on Comptel telecommunication company found out that introducing agile development was challenged by the culture and mind-set of the stakeholders [P05]. They further described instilling the agility culture requires long-term commitment, big investments, and customization to a company's specific situation. In organizations that had been using Traditional Software Development Methods (TSDM), which also called heavy-weight or plan-driven methodologies, trying to adopt agile scrum method at their core development processes have been challenged by the old mentality/culture of the need to have up-front documentation [P10].

Too many and Long Scrum Meetings. Agile framework consists of many ceremonies (daily standup meetings, product backlog grooming meetings, sprint retrospective, sprint review/demo, Release Iteration Planning sessions, Scrum of Scrum (SoS), Community of Practice (CoP), etc.). These meetings are perceived to less important and considered as waste of important software development time [P02]. For example, Practices like SoS had discovered to be inefficient in large projects [9] (as it was hard to find the right level of details in discussions to keep the forum interesting for many participants [P13]. It was reported that as the number of teams and its size increases, it becomes difficult to gather a large team for excessive meetings such as daily standup meetings [P04]. Bass identified appropriate and balanced agile ceremonies through studying artefacts from TSDM [P07].

As companies shifting from TSDM to Agile development, they certainly face different culture of development and practices, which is responsible to create the mind-set among people that agile contains a lot of unnecessary and long meetings [P17].

Improper implementation of Agile Processes. Many organizations think that they are agile because of the mere reason of adopting the framework without proper implementation knowledge and agile mind-set. To this end, a case study reports on the failure of Agile Scrum adoption in U.A.E. mainly due to improper implementation of the processes and lack of Agile coach who can guide the implementation [P17]. Furthermore, Bass's case study uncovered the absence of agile ceremonies for managing shared artefacts presents a challenge for practitioners [P07].

Lack of guidance from Literature. The successful implementation or adoption of Agile development methods demands for appropriate guidance from literature, which is hardly found [6]. For example, a longitudinal case study at F-Secure Corporation identified inefficient Release Iteration Planning because of lack of guidance from literature [P10]. Similarly, a case study Agile development and UX design teams coworking culture reported that due to the unavailability of literature that guides best collaboration mechanisms, many dependencies had happened between these two teams [P18].

 doi:10.20944/preprints202506.0080.v1

Agile Software Development (ASD) doesn't encourage reuse. ASD focuses on delivering single products for the customer. The method doesn't explicitly support the development of artifacts for reuse because agile discourages early upfront architecture and requirements documentations [P14]. Reusability of components is usually defined in the architecture, which is not sufficiently addressed in ASD [P07,P11]. To identify the possible Scrum tailoring to incorporate architectural extensions from Software Product Line (SPL), a case study was carried out on Large Financial IT Systems [P07]. Agile methods seem to be more suitable for new product development projects than maintenance projects, which tend to include legacy subsystems and monolithic functionality that can be hard to decompose into small independent pieces for agile software production [P14].

Agile Projects' Contract and Risk Management Issues. Agile methods emphasize individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [11]. The emphasis on individuals and interactions as well customer collaboration works well provided there are motivated and experienced agile developers together with customers who understand the core principles of agile. For example, the case study by Sundararajan & Bhasi [P12] on the risk management practices in large offshore-outsourced Agile Scrum software project, identified that agile scrum development was appropriate for such an environment provided "risks unique to such model are identified, assessed and appropriate risk resolution techniques deployed." In addition, as agile discourages upfront documentation, which may be required by regulatory authorities, particularly in financial sectors to forecast risks, agile development may pose serious challenges [P14]. Furthermore, architecture designers see agility to also contain risks and potentially challenging [P14].

Agile relies on self-motivated and highly empowered teams for a high-quality software production. When projects are completed, these important personnel are laid off if the company does not have projects in the pipeline. The high turnover of agile project team members has posed a serious a problem [P12,P14]. There is also critic that Agile favors team than rewarding high-performing individuals in the team. This results in high-caliber individuals to contribute less to the group [P12,P17]. In addition, Contract management of agile development estimates evolve and progress is not measurable in terms of earned value [P15,P17].

Project size. Many research work, experience, case study reports in the early adoption (2001 to 2010) of agile methodologies confirmed that it was useful for and successfully employed in small software development projects [6]. However, in recent years, agile methodologies are increasingly adopted in large-scale software development projects despite its shortcomings [12]. However, agile scrum applications for a team size exceeding 25 as well as when there are many teams and inter-teams interactions, is challenged [P01,P02,P04]. The optimal team size was suggested to be 5 to maximum 10, which ultimately enhances customers' feedback [P04]. When the back-end and front-end development is carried out by one scrum team, the amount of involved IT staff then easily exceeds the generally agreed upon maximum Scrum team size of 10 members [P09]. Conversely, a case study conducted by Riaz, Athar, and Buriro [P03] reported that they couldn't find any relationship between project size and its success. This calls for a further empirical study to accept their conclusion.

To address the project size issue, many Agile Scaling Frameworks have been introduced over the last eight years such as Scaled Agile Framework (SAFe) and Large Scale Scrum (LeSS) have been proposed by consultants and practitioners [5].

Agile practices adoption degree (Lacking Agile mindset). The degree of agile practices has direct impact on the success of agile projects [P03]. The practices adoption degree refers to perception of agility among project team members. The higher the perception of agility, the successful the project is. Some companies think that they are agile for the mere fact that they adopted the agile framework. This doesn't qualify them to be an agile company. It is the mind-set, practices, and adoption degree all together make a company agile [P01,P06,P10,P12,P13].

 doi:10.20944/preprints202506.0080.v1

Challenges in portfolio management. A company's software portfolio management penetrates multiple systems, software architectures, and features. Portfolio management is crucial for handling interdependencies and mitigating associated risks [P02]. Information Technology functionality in large companies is delivered by a portfolio of interdependent applications, not just a single application [13]. The iterative nature agile methods introduces new challenges in portfolio management that necessitate different patterns of action [P02]. Hence, agile teams lack portfolio-level understanding of projects they are involved in [P09]. It is extremely complex for an agile team to have a complete understanding of a company's portfolio unless product backlogs are created from it, where different products under one portfolio will be visible to all teams in different projects [P14]. For example, Bass's case study reported that there was comparatively little evidence of interaction between projects and the central release plan used for portfolio management [P07]. On the other hand, a case study carried out by Gren, Torkar, & Feldt reported that the application of Scrum adoption took longest within IT portfolio management [P06].

Over-optimism towards the New System. Agile methods don't warrant success in themselves per se but the mind-set of the people. Unless the mind-set of the people working in the software development is agile, the mere adoption of Scrum methodology doesn't bring the anticipated results [P17]. For example, a case study [P10] on a large-scale architecture development for an existing product project embracing many teams reported the over-optimism of the new software methodology, Scrum, made them to over-estimate the release iteration planning.

4.1.1.3. Multi-team Environment Challenges

Inter-team coordination Challenges. This has been identified among others as the most pertinent large-scale agile challenges demanding immediate research attention [14]. Coordination is defined as "managing dependencies between activities" [15]. Three classes of coordination have been discussed (1) mechanistic coordination – coordination by plan or rules with little communication, (2) organic coordination – coordination by means of mutual adjustment or feedback via interaction, which can be formal and planned or informal and spontaneous and (3) cognitive coordination – based on explicit and tacit knowledge the actors have about each other, such as a shared mental model [13].

In agile scrum development, group codependent teams are working on a project, which naturally creates dependencies and calls for effective coordination. Many software projects failure are associated with ineffective inter-team coordination [16]. To this end, many case studies on large-scale agile scrum development reported that inter-team coordination is the biggest challenge [P01,P02,P04,P05]. Out of the 18 cases identified in this study, 13 reported inter-team coordination is a huge challenge in large-scale agile scrum development, Appendix A exhibits specific cases.

Inter-team Communication Challenges. Effective coordination requires effective and appropriate communication. There are various definitions of coordination from a technical point view to organizational point of view [15]. In our context, we define Inter-team communication as how a codependent agile scrum teams can come to have "common knowledge" on software artefacts, which includes but not limited to common understanding of backlog items, release iteration planning, architecture, progress of other teams, and dependencies among features as well as software components. Failure to share and communicate these artefacts among agile Scrum teams were reported as challenges in large-scale software development. Of the 18 cases identified in this research, 14 reported this as a challenge [P01,P02,P03]. The full list of cases is exhibited in Appendix A. For example, the case study [P01] reported that there was a little communication or exchange about epics between the individual teams, inter-teams. The development teams were aware of other teams' general responsibilities but knew only little about their currently assigned epics [P01]. This made it difficult to clearly identify dependencies that happened in the inter-teams before the development begun.

Alignment Issues between teams and other stakeholders. Alignment issue is the difference in deadline between Scrum teams. When one teams submitted what was assigned to it while the others had not finished yet, the work of the finished team waited until the other team's work was completed

 doi:10.20944/preprints202506.0080.v1

to start feature testing [P09]. Although Scrum has a prescribed structure, the working processes can be implemented in many ways, leading to such misalignments. Definition of "done", which defines the work completed, is another factor that creates misalignment between teams. For example, one team define it as delivered before system testing and another team defines the same as delivered including system testing. Furthermore, misalignment between teams happened was due to misalignment of the start, the finish and duration of the sprints. One Scrum team has a two-weeks cycle and another team has a monthly cycle. Again, another alignment issue was the misalignment of test activities and test results between Scrum teams [P01,P02,P09]. The misalignment between codependent Scrum teams causes unpredictability and delivery delays [P02]. For instance, if the sprint of team B ends two weeks later than team A the delivery of the feature is delayed until team B had completed its sprint.

The misalignment between on-site customers and other stakeholders' goals, among the on-site customers, and between on-site customer and construction (Developers) was observed in the case study carried out by [P02]. These stakeholders' different priorities and goals created misalignments [P03,P10]. Therefore, there is a need to align activities at the program-level through Combining Product Backlogs and End-to-End Representation in Team [P15].

Geographic Distribution Challenges. In large-scale global software development, teams and their members are composed of from geographically distributed locations. This has created a challenge in the adoption and use of Agile development, which encourages face-to-face and collocated teams and its members. Teams in geographically dispersed locations have differences in cultural, time zone, language and ethnic group, which have created coordination and communication challenges [P04,P06,P07,P09,P12,P13,P15]. The camaraderie or team identity, and hold each other accounted, were problems within the distributed teams [P06]. Virtual teams and members were reported to be reluctance to accept responsibilities [P04,P06,P18]. In addition, the decisions are unclear among the virtual team members [P04,P06].

Unpredictability of delivery to commitment. One of the key predictability issues is the development of a single software package, by multiple Scrum teams in parallel, which makes it complex to correctly and collectively deliver the promised feature or product [P02,P04,P09,P18].

Group Maturity Issues. The agile methods trusts autonomous and empowered self-organizing individuals in a team than rigorous processes [11]. It also presumes self-reliant and experienced developers in a team. However, from the group development point of view, the team can't fully be autonomous. For example, the case study [P06] identified that there was a need from the team to be managed at the beginning.

The fact that a team is labeled as an agile team and doing agile practices does not mean it is a high performing team, that depends on the group maturity [P06]. Group development was found to be most important for better Performance, as defined in Social Psychology, [P06,P08,P09].

Visibility of Product Backlog and Operations Issues. The case study on three cases (telecommunications, insurance, and retail banking) [P09] reveled that a lack of information visibility in the chain (The lack of information visibility about the status and progress) as well as lack of supporting IT tools that could provide such visibility were perceived as serious issues in large-scale agile scrum teams. In general, lack of all backlog items, overall operational visibility and transparency front- to back-end were challenges in Agile Scrum [P07,P09,P10].

Collaboration Challenges. Agile Scrum teams should be self-organizing and empower to develop a set of user stories or requirements in a time-boxed time frame called sprint [11]. Members in the team are encouraged to work in an open environment to encourage collaboration and improve communication [6,9,17]. In additional, other scaling practices have been experimented and recommended for large-scale agile development's inter-team collaboration, such as CoP [17] and SoS [18]. The inter-teams collaboration, particularly back- to front-end teams collaboration had been report as a challenge [P0,P04,P09,P15].

 doi:10.20944/preprints202506.0080.v1

4.1.1.4. Challenges of Requirement Engineering

Requirements volatility and interdependencies. Requirements are volatile and subject to change [P01]. Often than not requirements are interdependent each other. It is paramount importance to recognize the volatility and interdependence nature of requirements. Case studies reported that these behavior of requirements make it challenging for agile teams to operate autonomously [P01][P02][P08][P09][P10].

Lack of Software Architecture Solutions Description/Documentation. Agile is a light-weight iterative development framework, which discourages huge upfront documentations. On contrary to this, in large-scale, mission-critical, safety-critical software development, it is strongly recommended that the architecture should clearly documented be documented upfront [P01][P02][P07][P08][P11][P13][P14]. "Agile Software Development does not promote formal documentation that may be required for regulatory, company policy and maintenance reasons within the financial services community" [P09]. Agile doesn't precisely prescribe to allocate time for architecture planning [P10].

Misalignment of Specification. In Large-Scale Agile Scrum development high-level also called epic, which is coarse grained requirements/product and release backlogs/ , are prepared by product owner and the low-level fine grained sprint backlogs are further drilled down into detail user stories by teams. The product owners may not have technical knowledge to identify the potential dependencies at the high-level requirements and teams have the technical knowledge but don't have access to and visibility of product as well as release backlog items. This lead to misalignment of specifications and inter-team dependencies [P01][P08]. In addition, misalignment happened because of multiple styles of documentation [P04].

Misalignment of Backlog Prioritization. This makes each Scrum teams concurrently developing different functionality for different features, which impedes the delivery of a feature at the end of the sprint. When backlog items are prioritized by a product owner without synchronizing the task with other product owners' goals resulted in mismatch between front- to back-end chain [P01][P02][P08][P09][P14][P18][P15]. In addition, there was case study reported in which the explicit criteria for requirements prioritization was lacking [P10].

Misalignment of Effort Estimation. This happens when high-level coarse-grained product and release backlog items are estimated without involvement of teams [P01][P04][P07]. This invisibility of effort estimation prevented development teams from knowing exactly when to expect handovers from other teams [P01][P12][P15]. Besides, unrealistic scope and of the project led to misaligned effort estimation [P10].

Misaligned Planning. This happened when agile teams plan the next iteration, also called Sprint in isolation with other teams, which created inconsistency in delivery of features, particularly back to front end-to-end development [P01][P02]. A case study [P09][P10] a lack of misaligned planning and communication between teams had been reported as a serious issue.

Unclear Decision Among Team Members. The decision among the virtual members of the team was not every time communicated and made clear to other core teams. This resulted in confusion among and between team members [P04]. Involvement of every member of the team and clear communication was reported as a useful technique to mitigate this problem [P06].

Misalignment of Task Allocation. When task allocations are not aligned across single teams (and central institutions) to provide all parties with the same understanding of dependencies, the teams will find themselves difficult to identify who they depend on [P01][P04][P08].

Customer Involvement Issues. In large-scale agile scrum development, many case studies reported that the involvement of large number of customers made it challenging to maintain the autonomy of teams in-line with the agile philosophy as well as increases the risk of failure because of the challenge of establishing a common understanding among all on-site customer representatives [P01][P02][P04][P08][P09][P10][P12][P15]. Another case studies [P04][P09][P10][P12][P15] confirmed that customer feedback and the participation of members in a team was largely affected by the team size.

 doi:10.20944/preprints202506.0080.v1

Requirements Management in Waterfall Mode. There is a tendency to manage software requirements as it used be managed in waterfall by Project Managers. This is against agile principles, which states the team should be free to select the one they want to develop. Case studies as in [P10][P15] observed this as a challenge in large-scale agile scrum development.

Issues of release planning. The release planning specifies how many user stories or features should be fully developed before a release. The challenges reported here is in the equation of identifying the optimal number of sprints/iterations in a release [P02] [P10]. Again, the successful release plan was highly dependent on the identification and handling of risky or conflicting feature development plans [P10]. In addition, increased configuration management effort due to an increased number of releases was reported [P09]. It was suggested that releasing planning to be guided by hiring external consultant to improve the processes [P15].

4.1.1.5. Knowledge Issues

Ineffective Leadership. Agile promotes organic leadership-flexible and participative encouraging cooperative social action is aimed at small and medium-sized organizations [12]. Applying organic leadership in large-scale agile scrum development teams where hundreds and thousands stakeholders are involved had been reported a huge challenge [P01][P02][P03][P15]. Hence, mix of mechanistic and organic leadership style had been suggested. However, the adoption of mix of these leadership styles were challenged due to lack of continuous improvement as well as lack of training and support during the adoption [P05].

Tacit knowledge management challenges. Agile development emphasis people than processes. This implies individual knowledge and experiences more important than formal defined and rigorous processes. However, tacit knowledge management had extremely been reported as a challenging venture in large-scale agile scrum development [P02][P06][P08][P11][P12][P13]. For example, some specialized highly specialized skills couldn't be shared easily[P09].

End-to-end implementation knowledge Issues. Often than not front- and back-end specialized agile teams work independently as reported in the case studies. Knowledge, Skills and Competency level of the Project Stakeholders lacked end-to-end feature(s) implementation knowledge, which led to dependencies and misalignment between teams [P01][P02][P08][P10][P12].

Fragmented view of the system. Lack of common understanding, holistic, or fragment view of the system among all customer representatives and the system stakeholders created misalignments among teams and challenged predictable feature(s) delivery. It was reported to be one source of dependencies between teams and features [P02][P05][P09][P10][P14][P18]. For example, in a case study [P10], stakeholders who participated in the release iteration planning session thought that they were drafting a plan that would serve until the next iteration planning (after three months). However, the idea was as much as possible to have the best plan that could be used as a base and would be modified within its timespan (three months). The stakeholder had no unified view of the system to understand what was meant by initial iteration planning.

Proof of Concepts with Core Technology. This highlights the need for experiments with new technologies before they are actually chosen and being used in the production of a system under consideration. By experimenting with the concepts with core technologies, a development team is familiarizing itself with the tools, methods, techniques and applications to succeed in these new technologies. As clearly mentioned in [P02], Flex Technology that was acquired for GUI layer had not been clearly understood by the developers for it to be used effectively and establish sound application architecture guidelines. The lack of resource allocation for concept proofing with new core technologies is also mentioned as a challenge, ibid. In a bid to avoid surprises, proof of concept prototypes had been employed to confirm architecture functionality [P14].

4.1.1.6. Resistance to Change

Skepticism towards the new Software Method. The fast adoption and improper implementation of the agile method resulted in a decrease in developers' productivity, which in turn

 doi:10.20944/preprints202506.0080.v1

created resentment and skepticism among the development teams [P17]. Hence, the team resorted to the previous waterfall method.

Agile promotes independent team working in time-boxed and is shielded from external influences. If teams have a high degree of independence, it results in a challenge to collaborate between enterprises, hence, the need for a chain of codependent teams to have a standard way to manage similar work while allowing local variations [P09]. Without this adjustment, there will be resentment by enterprises to adopt agile.

Management Resistance to Change. The top-level management commitment to change is of paramount importance for a smooth transition from water to the new methods, agile. The upper management concerns on the effectiveness and success of the transition to a new method is one of the stumbling blocks for agile transition [P17]. Agile can not thrive in within the organization without getting top-level management buy-ins. Their hesitation and resistance to resort to the new methods is a hiccup for agile adoption [P15].

4.1.1.7. Organizational Structures and Boundaries

Traditional IT Organizational Structure. With traditional IT organizational structure, which is centralized IT department, ensuring business involvement is the most challenging part of the agile adoption process [P15]. Because of the centralization of the IT department, the top-level management wants to keep control and assigns the Product Owner role to IT employees rather than business stakeholders [P12]. This creates havoc and less urgency among business representatives as they are not part of the development team [P15]. This coupled with the still presence of traditional project organization, which demands a marketing brief, project brief and written project initiation, consumes development time.

Superfluous Old Organizational Processes. Traditional large organizations have mechanistic organogram, which are rigid and bureaucratic with high formalization, and institutionalized processes that are poor fit to agile development [P13]. Agile requires an organic structure that is flexible and participative. It demands encouraging cooperative social actions. Codified rigid organizational processes are reported as the stumbling block for a successful agile implementation [P13] [P17].

4.1.1.8. Quality Assurance Challenges

Compromised Quality With the Smaller Releases. As the size and distribution of team increases, it is difficult to control the quality of documentation standards and design features [P04]. This same reference mentioned the size of the team is directly proportional to the Quality Assurance (QA) teams' support ability. The bigger the size and distribution of teams, the tougher it is for the QA to support the entire development team. This makes it challenging to accurately estimate developers' and QA's efforts on the onset of a certain project. With larger development and QA teams, iterative development such as agile development creates collaboration and coordination challenges, hence, the quality of minor smaller releases is compromised.

The 3C (Collaboration, Coordination, and Communication) challenges have been encountered by large and distributed development and QA teams, which in turn contributed to compromised quality in smaller releases [P09]. In addition, continues integration related issues have been identified [P10] "(1) the sensitivity of integration mistakes by the packaging maintainer which blocks the integration automation and (2) developers using own testing environments which are incompatible with the integrated testing environment, blocking the integration test." The dauting and challenging environment to realize continuous testing regimes using agile approaches is cited as a serious concern with smaller and continues releases [P15].

Lack of Automated Testing. It is researched that exploratory and collaborative testing are the essential factors in agile development. With a larger team, it will be unmanageable to keep the team in collaboration so that it performs well [P04]. Test effort and coverage are key challenges in agile

 doi:10.20944/preprints202506.0080.v1

scaling practices [P09]. Lack of an IT chain process automation to support a codependent chain of scrum team is identified as one of agile adoption issues [P09].

5. Discussions

Eighty-nine percent (16 out of the total eighteen cases) of the papers identified Agile difficult to implement as the critical challenges for harnessing agile Scrum for large-scale software development projects. 33% of these cases unveiled that lack of agile mindset deterred adoption degree of Agile practices while 28% of them found old software culture to move to Agile and Lean, too many and long Scrum meetings, project size and challenges in portfolio management are the primary reasons for failing to fully utilize agile Scrum for large-scale software development projects. Four cases (22%) are associated with Agile projects' contract and risk management issues. It is also interesting to find out that improper implementation of Agile processes and lack of guidance from literature, 11%, are among the challenges of adopting Agile Scrum for large development projects. ASD (Agile Software Development) doesn't encourage reuse, 17%, discovered in this main category.

Following agile difficult to implement challenges, the multi-team environment challenges constitute 83% (i.e. 15 out of eighteen reviewed papers' cases). Inter-team communication challenges (78%, 14 cases), inter-team coordination challenges (72%, 13 cases), and geographic distribution challenges (50%, 9 cases), and 33% (6 cases) discovered alignment issues between teams and other stakeholders as hiccups for full realization of agile Scrum for large software development projects. Similarly, seventy-eight percent (14 out of 18 cases) found out that challenges of engineering requirement are the scaling agile Scrum issue. Of this theme, lack of software architecture solutions description/documentation (50%-9 cases) unveiled as the main challenge in harnessing Agile Scrum for large-scale software projects. So do misalignment of backlog prioritization and customer involvement issues, each account for 44% (8 cases), under this substantive theme. Furthermore, misalignment of effort estimation as well as requirements volatility and interdependencies are among the top identified challenges, each accounting for 33% and 28%, respectively. Misaligned planning and issues of release planning equally contribute to the complexity, 22% cases, to the matter.

Knowledge Issues,13 papers out of 18 reviewed journal articles, i.e. 72%, is the third top-ranked theme. Of the subthemes under this category, tacit knowledge management challenges, which was identified in seven cases (39%), takes the first row. Fragmented view of the system 33% (6 cases) contributes to the knowledge issues. Ineffective leadership and end-to-end implementation knowledge issues equally contribute, 28%, to this substantive theme. Proof-of-concepts with core technology, 11%, found to be one of the identified hurdles in successfully harnessing Agile in large software development projects.

The overlapping and interdependent nature of software features created a dependency issue in agile development. To this end, 12 (67%) cases of the reviewed papers' cases found dependency issues. General lack of task dependency awareness is cited in 8(44%) cases under this category. Furthermore, expertise as well as technical dependencies equally, (4 cases-22%), orchestrate to the deterring adoption of agile. Similarly, business process and historical dependency, entity dependency, and social Loafing each contributing 11% to the issue.

Quality assurance challenges (Compromised Quality with the Smaller Releases and lack of automated testing) and organizational as well as boundaries (traditional IT organizational structure and superfluous old organizational processes) each account for 22% of their respective themes. In addition, resistance to change in which skepticism towards the new software method and management resistance to change subthemes are found to be the contributing factors for agile adoption reservation in the software development industry.

5.1. Limitations of the research

In the inclusion and exclusion criteria, the researchers' bias might have influenced the selection of primary studies, as well as data extraction. We used number of citations and language of publications as the delimiting criteria for inclusion and exclusion. In so doing, we might have

 doi:10.20944/preprints202506.0080.v1

excluded important journal articles. The number of citations is considered to select papers that have attracted the attention of industry and researchers alike. This, we believed, unveils the true challenges of adopting agile for large software development projects. Publication language was set to english only because of the researchers' language skills. There might have been important research papers excluded because of this.

Another challenge in the review processes was the limitations of Boolean keyword searches in online databases. It took iterations of keyword searches tests before landing on the ones identified in the faces.

6. Conclusions

The systematic review uncovered that agile Scrum has been widely adopted by large-scale software development projects, but with challenges and limitations. The review identified eight substantive themes and 46 subthemes under each of these main themes from eighteen full journal article reviews. Agile difficult to implement is found to be a stumbling block for adopting Agile Scrum for large-scale software development projects. Lack of agile mindset, the high frequency of Scrum meetings coupled with improper implementation of Agile processes, challenges in portfolio management, and project size challenged the adoption of the method. When the project and team size increases, companies find it challenging to fully realize Scrum. Nonetheless, today's large-scale software development projects calls for communication and collaboration of diverse teams from across the globe with significantly varying time zones. "Agile Projects' Contract and Risk Management Issues" happen because agile emphasis team interaction over rigid processes, collaboration over contractual agreements and discourages upfront documentation. Some projects inherently require the identification of risks way before the actual development kicks off. This seriously challenges the agile Scrum tenets. Hence, there is a need to tailor the method in abid to make it fit for large-scale development endeavors. In this case, the risks should be clearly articulated and experienced developers who can quickly understand and adjust according to business needs. Improper implementation of the agile processes by businesses is another staggering challenge for agile Scrum. In addition, old software culture to move towards agile and lean, too many and long Scrum meetings, and over-optimism, taking agile Scrum as "a silver bullet", are another stumbling block for agility. Furthermore, the multi-team development environment of today's global software practices makes inter-team coordination, collaboration and communication a dauting task. The invisibility of product backlogs to all teams across the globe participating in the development created hiccups. The requirement engineering challenges such as lack of upfront architectural documentation, fragility of agile requirements, misalignment of prioritized backlogs, customer involvement issues and others are orchestrators to the challenge. Tacit knowledge management, fragmented view of the system, and others mentioned in the subthemes contribute to the knowledge management challenges of agile Scrum.

7. Recommendations for future Studies

The following points enrich and solidify the findings of the present research.

The researchers considered only eighteen full journal articles for review. However, considering more empirical papers will enrich the findings of this paper.

Considering journal articles published in other languages other english language might give more context.

Appendix A

Appendix A.1

Table A1. Challenges of Using Agile Scrum for Large-Scale Software Development.

 doi:10.20944/preprints202506.0080.v1

Challenge Type	Primary Sources	Case Organization	#of cases
Dependency Issues (12)-67%			
General Lack of Task Dependency Awareness	P01,P02,P04,P07,P08,P09,P10,P15	C01,C02,C04,C07,C08,C09,C10,C15	8(44%)
Expertise dependency	P08,P09,P11,P12	C08,C09,C11,C12	4(22%)
Business process and Historical dependency	P08,P11	C08,C11	2(11%)
Entity Dependency	P08,P18	C08,C18	2(11%)
Technical Dependency	P08,P14,P15,P18	C08,C14,C15,C18	4(22%)
Social Loafing	P04,P10	C04,C10	2(11%)
Agile Difficulty to implement (16)-89%			
Old Software Culture to move to Agile and Lean	P05,P06,P10,P15,P17	C05,C06,C10,C17,C15	5(28%)
Too many and Long Scrum Meetings	P02,P04,P07,P13,P17	C02,C07,C04,C17,C13	5(28%)
Improper implementation of Agile Processes	P07,P17	C07,C17	2(11%)
Lack of guidance from Literature	P10,P18	C10,C18	2(11%)
ASD (Agile Software Development) doesn't encourage reuse	P07,P11,P14	C07,C11,C14	3(17%)
Agile Projects' Contract and Risk Management Issues	P12,P14,P15,P17	C12,C14,C15,C17	4(22%)
project size	P01,P02,P03,P04,P09	C01,C02,C03,C04,C09	5(28%)
Agile practices adoption degree (Lacking Agile mindset)	P01,P03,P06,P10,P12,P13	C01,C03,C06,C10,C12,C13	6(33%)
challenges in portfolio management	P02,P06,P07,P09,P14	C02,C06,C07,C09,C14	5(28%)
Over-optimism	P10,P17	C10,C17	2(11%)
Multi-team Environment Challenges (15)-83%			
Inter-team coordination Challenges	P01,P02,P04,P05,P06,P07,P08,P09,P10,P13,P14,P15,P18	C01,C02,C04,C05,C06,C07,C08,C09,C10,C13,C14,C15,C18	13(72%)
Inter-team Communication Challenges	P01,P02,P03,P04,P05,P06,P07,P08,P09,P12,P13,P14,P15,P18	C01,C02,C03,C04,C05,C06,C07,C08,C09,C12,C13,C14,C15,C18	14(78%)
Alignment Issues between teams and other stakeholders	P01,P02,P03,P09,P10,P15	C01,C02,C03,C09,C10,C15	6(33%)
Geographic Distribution Challenges	P04,P06,P07,P09,P10,P12,P13,P15,P18	C04,C06,C07,C09,C10,C12,C13,C15,C18	9(50%)

 doi:10.20944/preprints202506.0080.v1

unpredictability of delivery to commitment	P02,P04,P09,P18	C02,C04,C09,C18	4(22 %)
Group Maturity Issues	P06,P08,P09	C06,C08,C09	3(17 %)
Visibility of Product Backlog and Operations Issues	P06,P07,P10	C06,C07,C10	3(17 %)
Collaboration Challenges	P03,P04,P09,P15	C03,C04,C09,C15	4(22 %)
Challenges of Requirement Engineering (14)-78%			
Requirements volatility and interdependencies	P01,P02,P08,P09,P10	C01,C02,C08,C09,C10	5(28 %)
Lack of Software Architecture Solutions Description/Documentation	P01,P02,P07,P08,P09,P10,P11,P13,P14	C01,C02,C07,C08,C09,C10,C11,C13,C14	9(50 %)
Misalignment of Specification	P01,P04,P08	C01,C04,C08	3(17 %)
Misalignment of Backlog Prioritization	P01,P02,P08,P09,P10,P14,P18,P15	C01,C02,C08,C09,C10,C14,C18,C15	8(44 %)
Misalignment of Effort Estimation	P01,P04,P07,P10,P12,P15	C01,C04,C07,C10,C12,C15	6(33 %)
misaligned planning	P01,P02,P09,P10	C01,C02,C09,C10	4(22 %)
Unclear Decision Among Team Members	P04,P06	C04,C06	2(11 %)
Misalignment of Task Allocation	P01,P04,P08	C01,C04,C08	3(17 %)
Customer Involvement Issues	P01,P02,P04,P08,P09,P10,P12,P15	C01,C02,C04,C08,C09,C10,C12,C15	8(44 %)
Requirements Management in Waterfall Mode	P10,P15	C10,C15	2(11 %)
Issues of release planning	P02,P09,P10,P15	C02,C09,C10,C15	4(22 %)
Knowledge Issues (13)-72%			
Ineffective Leadership	P01,P02,P03,P05,P08	C01,C02,C03,C05,C08	5(28 %)
Tacit knowledge management challenges	P02,P06,P08,P09,P11,P12,P13	C02,C06,C08,C09,C11,C12,C13	7(39 %)
End-to-end implementation knowledge Issues	P01,P02,P08,P10,P12	C01,C02,C08,C10,C12	5(28 %)
Fragmented view of the system	P02,P05,P09,P10,P14,P18	C02,C05,C09,C10,C14,C18	6(33 %)
Proof-of-concepts with core technology	P02,P14	C02,C14	2(11 %)
Resistance to Change (3)-17%			
Skepticism towards the new Software Method	P09,P17	C09,C17	2(11 %)
Management resistance to Change	P15,P17	C15,C17	2(11 %)
Organizational Structure and Boundaries (4)-22%			

 doi:10.20944/preprints202506.0080.v1

Traditional IT Organizational Structure	P12,P15	C12,C15	2(11 %)
Superfluous Old Organizational Processes	P13,P17	C13,C17	2(11 %)
Quality Assurance Challenges (4)-22%			
Compromised Quality with the Smaller Releases	P04,P09,P10,P15	C04,C09,C10,C15	4(22 %)
Lack of Automated Testing	P04,P09	C04,C09	2(11 %)

Appendix B

Appendix B.1

Table A2. Categorization of Papers for Review Based on Context Code.

Study Type (C-case study, ER-Experience Report, Empirical Study-ES)	Paper (s)	Company	Business area	Software Organization Size	Operating Locations
C01	P01	Anonymous	Enterprise Software Solutions	13 dev teams (Each consists of 6 to 16 developers) and total of over 140 Employees	China, India and Germany
C02	P02	Norwegian Public Service Pension Fund (the "Pension Fund") Developed by the Pension Fund internal development unit, and Accenture and Steria Consulting Firms	Automation of Public Service Pension Fund	12 collocated development teams with more 175 people involved. The development ran for four years.	Norway
C03 (ES)	P03	Anonymous			Pakistan
C04 (ES)	P04	Anonymous			Researchers from Pakistan

 doi:10.20944/preprints202506.0080.v1

C05 (ER)	P05	Comptel	Telecommunications	750+ employees	Finland
C06 (ER)	P06	Anonymous			Different places
C07 (ES)	P07	Anonymous			Companies involved in the study were based in U.K. and India
C08	P08	Anonymous	Government, Commercial service provider, and Commercial software development firm	Three companies with 2000, 200, and 20 employees, respectively	New Zealand
C09	P09	Anonymous	Retail Banking, Telecommunications, Insurance	150 Scrum Teams, 34 Scrum Teams, and 5 Scrum Teams, respectively. Each of this case organizations have from 250 to 1500 IT development employees.	Netherlands
C10	P10	F-Secure	PC, Mobile and Data Security	More than 800 Employees	Finland (Development teams from Poland and Malaysia were involved)
C11 (ES)	P11	Anonymous			
C12	P12	Anonymous	Healthcare provider in the USA	Six Agile Scrum Teams (Each contains six to seven developers)	The Development was carried out in three locations (U.S.A. (Owner of the Project), Chennai and Bangalore)
C13	P13	Ericsson	Telecommunication	400 persons in 40 Scrum teams at three sites	Finland, Hungary and the US

 doi:10.20944/preprints202506.0080.v1

C14	P14	Anonymous	Financial	Very big Financial IT Systems	Not mentioned
C15	P15	Anonymous	Financial and Manufacturing	26,000 and more than 100,000 employees, respectively	Netherlands
C16	P16	Anonymous	Rapid application development and production tool	More than 20,000 employees and has clients in 70 countries	Not mentioned
C17 (ER)	P17	Anonymous	Telecommunications	More 200	U.A.E
C18	P18	Anonymous	Media Organization	Not mentioned	Not mentioned

Appendix C

Appendix C.1

Table A3. Categorization of Papers for Review Based on Context Code of Research Processes.

Case	Study Focus	Key Results the Study Focuses On	Subject
C01	Coordination Challenges in Large-Scale Software Development	Planning misalignment leads to lack of dependency awareness, which ultimately leads to coordination challenges	23 semi-structured interviews with key informants and analysis of documents.
C02	Adoption of Agile Scrum in Large-Scale Software Development	customer involvement, software architecture, and inter-team coordination were identified as key challenges	group interviews with 24 participants and documents
ES03	Social Success Factors Affecting Implementation of Agile Software Development Methodologies	visionary leadership, degree or level of Agile software practices, congruence value, etc. were identified as significant contributors to the success of a project	Interviews with 271 software professionals representing 28 companies
ES04	Preference of Using Agile Scrum with Large Team size	Agile scrum is preferred to be used when the team size is less than 25	Survey data was collected from several software houses in a developing country

 doi:10.20944/preprints202506.0080.v1

ER05	Adoption of SAFe (Scaled Agile Framework) in two business lines of Comptel	Key areas for successful adoption of the agile framework	Interviews with key informants
ES06	Group development and group maturity when building agile teams	group developmental aspects is key factors to a successful agile transition	Ten semi-structured Interviews with individuals from four companies and 66 Surveys from another four companies, a total of eight companies
ES07	Identifying Agile Artefacts to enrich the traditional Agile Ceremonies	By way of Identifying Agile Artefacts additional Agile Ceremonies were identified	46 practitioner interviews, documentary sources and observations, in nine international companies
C08	Taxonomy of dependencies to apply appropriate coordination in Agile	Identified the key dependency taxonomy and Suitable Agile coordination	11 interviews (project leader, developer, business analyst, domain expert, or tester)-Each Interview lasted 40 to 90 min following a semi-structured interview schedule.
C09	Identify the collaboration related issues in chains of Scrum teams	governance framework to manage chains of Scrum teams	Three case studies (9, 6, 3 Product Owners, Line Managers and Scrum Masters were interviewed from each case, respectively).
C10	Release Iteration Planning in Agile Development	Benefits and best practices of Release Iteration Planning in Agile	Two Case Studies (Interviews and Observation)
ES11	Case Based Reasoning (CBR) in Agile Scrum Development	CBR to learn from past projects in Agile Scrum is useful	Experiment and survey (27 developers from two companies)
C12	risk management in a large offshore-outsourced Agile Scrum software development	Risk identification and mitigation techniques in a large offshore-outsourced Agile Scrum Software Development	1 Offshore Development Center Project Manger and 2 Scrum Masters)
C13	Community of Practice (CoP) to help improve Agile Adoption	CoPs is supporting continuous organizational improvements	52 semi-structured interviews on two sites
C14	studying the adoption and architectural extensions of the Scum method in large Financial IT Systems	Methods and Practices to adopt Agile Scrum in Financial IT Sector	21 interviews, including 33 specialists from the case organization's different areas and managerial levels (each interview lasted 1.5-2 hours) and workshops were organized

 doi:10.20944/preprints202506.0080.v1

C15	Challenges and Remedies of Coexistence of Plan-driven and Agile Methods	Suggested mitigation Strategies to coexist the two methodologies.	Interviews with 21 Agile Practitioners from two Large Enterprise Organizations (duration of Interview 1 to 1.5 hours)
C16	onside customer involvement in Agile Scrum development	Scrum can be successfully used without intensive involvement of onsite customer	Interview, Observation, Survey, and Document analysis.
ER17	Agile Scrum Adoption Strategy	Agile Scrum Adoption Failure, case study	Experience report on Agile Scrum Adoption
C18	User Experience (UX) Design and Agile Method	cooperation between the Agile developers and UX designers was achieved through ongoing articulation work by the developers	Observation of and interview with 14 Agile Developers and 2 UX Designers

Appendix D.

Appendix D1. List of References for the Review

P01. Bick, S.; Spohrer, K.; Hoda, R.; Scheerer, A.; Heinzl, A. Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. *IEEE Transactions on Software Engineering* 2018, *44*, 932–950, doi:10.1109/TSE.2017.2730870.

P02. Dingsøyr, T.; Moe, N.B.; Fægri, T.E.; Seim, E.A. Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation. *Empir Softw Eng* 2018, *23*, 490–520, doi:10.1007/s10664-017-9524-2.

P03. Riaz, M.N.; Mahboob, A.; Buriro, A. Social Success Factors Affecting Implementation of Agile Software Development Methodologies in Software Industry of Pakistan: An Empirical Study. *International Journal of Advanced Computer Science and Applications* 2018, *9*, doi:10.14569/IJACSA.2018.090713.

P04. Zia, A.; Arshad, W.; Mahmood, W. Preference in Using Agile Development with Larger Team Size. *International Journal of Advanced Computer Science and Applications* 2018, *9*, doi:10.14569/IJACSA.2018.090716.

P05. Ebert, C.; Paasivaara, M. Scaling Agile. *IEEE Softw* 2017, *34*, 98–103, doi:10.1109/MS.2017.4121226.

P06. Gren, L.; Torkar, R.; Feldt, R. Group Development and Group Maturity When Building Agile Teams: A Qualitative and Quantitative Investigation at Eight Large Companies. *Journal of Systems and Software* 2017, *124*, 104–119, doi:10.1016/j.jss.2016.11.024.

P07. Bass, J.M. Artefacts and Agile Method Tailoring in Large-Scale Offshore Software Development Programmes. *Inf Softw Technol* 2016, *75*, 1–16, doi:10.1016/j.infsof.2016.03.001.

P08. Strode, D.E. A Dependency Taxonomy for Agile Software Development Projects. *Information Systems Frontiers* 2016, *18*, 23–46, doi:10.1007/s10796-015-9574-1.

P09. Vlietland, J.; van Vliet, H. Towards a Governance Framework for Chains of Scrum Teams. *Inf Softw Technol* 2015, *57*, 52–65, doi:10.1016/j.infsof.2014.08.008.

P010. Heikkilä, V.T.; Paasivaara, M.; Rautiainen, K.; Lassenius, C.; Toivola, T.; Järvinen, J. Operational Release Planning in Large-Scale Scrum with Multiple Stakeholders – A Longitudinal Case Study at F-Secure Corporation. *Inf Softw Technol* 2015, *57*, 116–140, doi:10.1016/j.infsof.2014.09.005.

P011. Turani, A. APPLYING CASE BASED REASONING IN AGILE SOFTWARE DEVELOPMENT. *J Theor Appl Inf Technol* 2015, *78*.

 doi:10.20944/preprints202506.0080.v1

P012. Sundararajan, S.; Bhasi, M.; Vijayaraghavan, P.K. Case Study on Risk Management Practice in Large Offshore-outsourced Agile Software Projects. *IET Software* 2014, *8*, 245–257, doi:10.1049/iet-sen.2013.0190.

P013. Paasivaara, M.; Lassenius, C. Communities of Practice in a Large Distributed Agile Software Development Organization – Case Ericsson. *Inf Softw Technol* 2014, *56*, 1556–1577, doi:10.1016/j.infsof.2014.06.008.

P014. Hajjdiab, H.; Taleb, A.S.; Ali, J. An Industrial Case Study for Scrum Adoption. *Journal of Software* 2012, *7*, 237–242, doi:10.4304/jsw.7.1.237-242.

P015. Ihme, T. Scrum Adoption and Architectural Extensions in Developing New Service Applications of Large Financial IT Systems. *Journal of the Brazilian Computer Society* 2013, *19*, 257–274, doi:10.1007/s13173-012-0096-0.

P016. van Waardenburg, G.; van Vliet, H. When Agile Meets the Enterprise. *Inf Softw Technol* 2013, *55*, 2154–2171, doi:10.1016/j.infsof.2013.07.012.

P017. Inayat, I.; Noor, M.A.; Inayat, Z. Successful Product-Based Agile Software Development without Onsite Customer: An Industrial Case Study. *International Journal of Software Engineering and Its Applications* 2012, *6*.

P018. Ferreira, J.; Sharp, H.; Robinson, H. User Experience Design and Agile Development: Managing Cooperation through Articulation Work. *Softw Pract Exp* 2011, *41*, 963–974, doi:10.1002/spe.1012.

References

1. Schwaber, Ken. Agile Project Management with Scrum. In; Microsoft Press, 2004 ISBN 9780735619937.
2. Schwaber, K.; Beedle, M. *Agile Software Development with Scrum*; Prentice Hall: Upper Saddle River, 2002;
3. Paasivaara, M. Adopting SAFe to Scale Agile in a Globally Distributed Organization. *Proceedings - 2017 IEEE 12th International Conference on Global Software Engineering, ICGSE 2017* **2017**, 36–40. https://doi.org/10.1109/ICGSE.2017.15.
4. Kasauli, R.; Knauss, E.; Kanagwa, B.; Balikuddembe, J.K.; Nilsson, A.; Calikli, G. Safety-Critical Systems and Agile Development: A Mapping Study. *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* **2018**, 470–477. https://doi.org/10.1109/SEAA.2018.00082.
5. Ebert, C.; Paasivaara, M. Scaling Agile. *IEEE Softw* 2017, *34*, 98–103.
6. Dikert, K.; Paasivaara, M.; Lassenius, C. Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review. *Journal of Systems and Software* **2016**, *119*, 87–108. https://doi.org/10.1016/j.jss.2016.06.013.
7. Kitchenham, B.; Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; Budgen, D., Brereton, P., Linkman, M.T.S., Mendes, M.J.E., Visaggio, G., Eds.; Version2.3.; Durham, UK, 2007;
8. Moher, D.; Shamseer, L.; Clarke, M.; Ghersi, D.; Liberati, A.; Petticrew, M.; Shekelle, P.; Stewart, L.A. Preferred Reporting Items for Systematic Review and Meta-Analysis Protocols (PRISMA-P) 2015 Statement. *Syst Rev* **2015**, *4*, 1. https://doi.org/10.1186/2046-4053-4-1.
9. Paasivaara, M.; Lassenius, C.; Heikkilä, V.T. Inter-Team Coordination in Large-Scale Globally Distributed Scrum. *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '12* **2012**, 235. https://doi.org/10.1145/2372251.2372294.
10. Cho, J.J. An Exploratory Study on Issues and Challenges of Agile Software Development with Scrum. *All Graduate Theses and Dissertations* **2010**, 599.
11. Beck, K.; Beedle, M.; Van Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.; Hunt, A.; Jeffries, R.; et al. Manifesto for Agile Software Development Available online: https://moodle2016-17.ua.es/moodle/pluginfile.php/80324/mod_resource/content/2/agile-manifesto.pdf (accessed on 1 October 2018).
12. Dingsøyr, T.; Moe, N.B.; Fægri, T.E.; Seim, E.A. Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation. *Empir Softw Eng* **2018**, *23*, 490–520. https://doi.org/10.1007/s10664-017-9524-2.
13. Vlietland, J.; Van Vliet, H. Towards a Governance Framework for Chains of Scrum Teams. *Inf Softw Technol* **2015**, *57*, 52–65. https://doi.org/10.1016/j.infsof.2014.08.008.

 doi:10.20944/preprints202506.0080.v1

14. Moe, N.B.; Olsson, H.H.; Dingsøyr, T. Trends in Large-Scale Agile Development: A Summary of the 4th Workshop at XP2016. In Proceedings of the Proceedings of the Scientific Workshop Proceedings of XP2016 on - XP '16 Workshops; ACM Press: New York, New York, USA, 2016; Vol. 1, pp. 1–4.
15. Malone, T.W.; Crowston, K. The Interdisciplinary Study of Coordination. *ACM Computing Surveys (CSUR)* **1994**, *26*, 87–119. https://doi.org/10.1145/174666.174668.
16. Bick, S.; Spohrer, K.; Hoda, R.; Scheerer, A.; Heinzl, A. Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. *IEEE Transactions on Software Engineering* **2018**, *44*, 932–950. https://doi.org/10.1109/TSE.2017.2730870.
17. Paasivaara, M.; Lassenius, C. Communities of Practice in a Large Distributed Agile Software Development Organization - Case Ericsson. *Inf Softw Technol* **2014**, *56*, 1556–1577. https://doi.org/10.1016/j.infsof.2014.06.008.
18. Sutherland, J. Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies. *Cutter IT Journal* **2001**, *14*, 5–11. https://doi.org/10.1093/glycob/cwq056.