Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review

Henry Edison, Xiaofeng Wang, and Kieran Conboy

Abstract—Following the highly pervasive and effective use of agile methods at the team level, many software organisations now wish to replicate this success at the organisational level, adopting large-scale agile methods such as SAFe, Scrum-at-Scale, and others. However, this has proven significantly challenging. An analysis of the extant literature reveals a disparate set of studies across each individual method, with no cross-method comparison based on empirical evidence. This systematic literature review compares the main large-scale agile methods, namely SAFe, LeSS, Scrum-at-Scale, DAD, and the Spotify model. It is the first study to analyse and compare each of the method's principles, practices, tools, and metrics in a standardised manner. For each method, it presents not just the original method specifications but also all extensions and modifications to each method proposed by subsequent empirical research. It includes in this comparison not just commercial large-scale methods but also those that have been custom-built in organisations such as Nokia, Ericsson, and others. Based on the findings reported in this study, practitioners can make a more informed decision as to which commercial method or method component or, indeed, custom-built method is better suited to their needs. Our study reveals a number of theoretical and practical issues in the current literature, such as an emphasis on the practices of commercial frameworks at the expense of their underlying principles, or indeed any of the custom method. A set of challenges and success factors associated with the use of large-scale agile methods are identified. The study also identifies a number of research gaps to be addressed across methods.

Index Terms—large-scale agile, critical assessment, challenges and success factors, systematic literature review

1 Introduction

FOR the last two decades, agile methods such as Scrum [1], [2], eXtreme Programming (XP) [3], DevOps [4], and flow [5] have proven overwhelmingly popular and, for the most part, highly effective amongst software development teams. More recently, attention has turned to the challenge of designing and implementing large-scale, organisation-wide variants of these methods. Reasons underpinning the emergence of these large-scale[1] methods include a need for alignment and cohesion across many teams, deep interdependencies between software development and other organisational functions such as human resources, legal, and finance, as well as the global trend toward large distributed teams and product delivery at scale [6]. This movement has been given many different labels such as 'large-scale development', 'process transformation' [6], [10] and 'organisation-wide transformation' [11]. Over the last number of years, quite a few large-scale agile methods have been proposed, e.g. the Scaled Agile Framework (SAFe) [12], and Large-Scale Scrum (LeSS) [13].

1. There is no firm agreement as to what constitutes large-scale development in the literature [6], [7]. Barlow et al. [8] define the term as characterised by multiple stakeholders and complex projects within large organisations. Rolland et al. [9] suggest that large-scale involves complex integration with various internal and external information systems, multiple stakeholders with different interests. The review by Dikert et al. [6] provides a more focused and quantifiable definition of large scale development, interpreting it as involving more than 50 developers or at least six teams working on a common product or project in the same organisation. This is the definition adopted in this study.

Previous research indicates that the scaling up of agile methods to large-scale development presents "a thorny set of issues" [15], such as the questionable assumption that effective large-scale development can be achieved by simply scaling up small-scale agile methods [9]. The implementation of large-scale methods, regardless of prescribed or scaled-up from team-level, has proven highly challenging, with few successful cases to date [16]. These scaled-up methods have struggled to deal with the exponentially greater complexities and interdependencies of large-scale, organisation-wide development. The difficulties usually centre around the complexities and ambiguities, arising from (i) a large number of teams, roles, and personalities; (ii) an often unknown composition of participant teams and projects at the outset; (iii) abstract, knowledge-intensive, and often ill-structured work processes; (iv) diverse and often competing agendas between teams that sometimes contradict the organisation itself; (v) abstract, complex, and often unknown final outputs and goals [9], [10], [16].

As is often the case with new and emerging phenomena in software development, practice has led research, with the creation, promotion, and dissemination of these large-scale agile methods largely due to the efforts of practitioners and consultants [16]. While this practice-driven nature certainly has merits, Rolland et al. [9] argue that the method and method implementations are often built on fundamentally incorrect assumptions. There is a need for a review that reflects and critiques the conceptual underpinnings of these large-scale methods.

A further problem is that what limited literature does exist tends to focus on single methods, such as SAFe, Spotify model or DAD [6], [7], [9], [17], [18], [19], rather than any comparison between them. Researchers and practitioners need some means of comparing and contrasting between these methods to identify the advantages and disadvantages of each, to choose which parts are most suited to their particular context, to identify the gaps of each method and to identify new principles, practices, tools, and metrics that can be used to fill those gaps. These actions will then hopefully further the body of knowledge of large-scale development and lead to more effective outcomes than at present.

1.1 Research Questions

To address the issues identified above, this study's objective is set to systematically identify and analyse empirical evidence of large-scale agile development methods in the literature. This study will specifically focus on a comparison between methods. The study addresses the following research questions:

RQ1 To what extent have large-scale agile methods been studied empirically in the literature?
RQ2 To what level of abstraction (*principles, practices, tools, and metrics*) are the large-scale agile methods identified in RQ1 studied empirically, and what are the resulting knowledge gaps?
RQ3 What challenges have been reported for the large-scale agile methods identified in RQ1?
RQ4 What success factors have been reported for the large-scale agile methods identified in RQ1?

1.2 Contributions of the Study

Aside from being the most up-to-date review of large-scale agile methods, including 191 primary studies across 134 organisations, this study makes a number of contributions to research and practice.

First, while others have also studied large-scale methods, this is the first to compare and contrast between the methods themselves (e.g. SAFe, Scrum-at-Scale, LeSS, DAD, and scaled agile methods). Others either focus on one method (e.g. [7], [20]) or they study the overall methods in a collective, aggregated manner that does not allow researchers or practitioners to compare and contrast these methods (e.g. [6], [8], [9], [18], [21], [22], [23]). This study is also the first to include custom-built methods (RQ1).

Second, the study is novel in that it conducts this comparison across a standard set of headings, namely each method's principles, practices, tools, and metrics [24]. A method comprises different levels of abstraction: A *principle* is a proposition that serves as the foundation or basis for a system. It does not prescribe an action or process but rather provides a basis for making those decisions. *Practices* are habitual or customary ways of doing something, acknowledged by a community as the correct way to do things [25], [26]. *Tools* (e.g. diagrams, notations, or computer support) are used to support the application of the practices [24]. *Metrics* are used to evaluate performance when using the method. This facilitates a structure for a balanced comparison and evaluation of each method's strengths, weaknesses, and gaps. This study then analyses the empirical studies of the large-scale agile frameworks under each of these headings. We identify and present various extensions to these large-scale agile frameworks that have been proposed in the literature (RQ2). Until now, original versions of each method were separated from any new recommended additions that emerged through various empirical studies. This allows the building of cumulative tradition, a positive trait of any research area where each piece of research can build on a solid foundation encompassing all existing research.

Finally, our study identifies the challenges (RQ3) and key success factors (RQ4) associated with the application of large-scale agile methods. Challenges are issues or obstacles that demand great consideration and need to be overcome [27]. When challenges are unmanaged, they may cause project delays, quality issues, or failure [28], [29]. Success factors are the things that "must go right" as they are strongly related to the achievement of strategic goals [30], [31]. Most existing literature reviews focus on either challenges or success factors only and study those of large-scale agile transformation in general. We incorporated both challenges and success factors across each methods. Including both aspects in one study enables us to produce the most comprehensive overview possible. Therefore if a researcher or practitioner is using this study to examine challenges and success factors, they have an over-arching set of all relevant issues, regardless of whether the original authors labelled them as a challenge or success factor.

1.3 Structure of the Paper

The remainder of this article is structured as follows. Section 2 presents the related work in this area. This is followed by description of our research approach in Section 3. In Section 4, we present the characteristics of the primary studies included in our review. The key findings of this study are presented in Section 5 and discussed in Section 6, along with the implications for research and practice. Section 7 then presents the conclusions of the study.

2 Related Work

Relevant systematic reviews were identified by searcing in the Compendex, ISI Web of Science, Scopus, and AIS e-Library digital databases. We used the following search string to search within title and abstract and keywords, combined with the synonyms of "systematic literature reviews" [32]: *(transform* OR transiti* OR migrat* OR journey OR "rollout" OR "large scale" OR scale* OR enterprise OR portfolio OR "mega project" OR "mega system" OR "systems-of-systems" OR distributed) AND (software OR "information system" OR "information systems") AND ("systematic review" OR "research review" OR "research synthesis" OR "research integration" OR "systematic overview" OR "systematic research synthesis" OR "integrative research review" OR "integrative review" OR "systematic literature review" OR "literature review")*.

This search string returned 1,677 papers in total. After removing duplicates, the titles and abstracts of the remaining articles were read. We identified 19 relevant systematic literature reviews in this area ([4], [6], [7], [9], [17], [20], [21], [22], [23], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42]). By reading the titles of articles that cite these reviews, we identified three more relevant articles ([18], [43], [44]). During the actual search process of primary studies, we also found another two relevant literature review studies ([8], [45]). In total, 24 related literature reviews were identified. The results of our assessment are discussed below.

The majority of the reviews are focused on the adoption of agile methods in (i) global software development (GSD) (9 reviews), (ii) software product lines (SPL) (2 reviews), (ii) a single method, e.g. Lean (2 reviews) and DevOps (1 review), and (iii) large-scale agile development (10 reviews). Except for the reviews by Hossain et al. [34] and Lous et al. [38], which studied Scrum, all reviews on GSD do not distinguish between methods in their analysis. GSD does not necessarily mean large; thus, these reviews also included studies on small-sized teams as their primary studies. The findings of these studies, e.g. challenges, barriers, or success factors were reported in an aggregated manner and did not provide method-specific information. This is also the case of the agile and SPL studies [33], [37] or Lean [36], [44]

Ten reviews on large-scale agile development were considered most relevant to this study ([6], [7], [8], [9], [18], [20], [21], [22], [23], [43]). In terms of the review method, only five out of ten reviews were guided by well-known literature review approaches ([6], [7], [21], [22], [23]). Two reviews employed a more ad-hoc approach ([9], [18]). Two reviews did not report their review method ([8], [43]), while one used a multi-vocal approach that combined peer-reviewed and grey literature. The studies of Abrar et al. [21] and Dikert et al. [6]) present extensive literature reviews by following the guidelines by Kitchenham and Charter [46] but do not perform a quality assessment.

In terms of the large-scale agile methods studied (RQ1), seven reviews [6], [8], [9], [18], [21], [22], [23] focused on the adoption of team-level agile methods in large-scale projects. Three reviews studied the commercial large-scale methods, e.g. SAFe [20], SAFe and LeSS [7], and DAD, SAFe, LeSS, Spotify, Nexus, RAGE [43]. However, the study by Alqudah & Razali [43] did not provide empirical evidence as it mixed the findings from empirical research and seminal books or articles. In terms of the level of abstraction (RQ2), the review of Kalenda et al. [7] reported studies that examined the practice level of the methods in an aggregated way, rather than method-specific. In terms of challenges and success factors (RQ3 & RQ4), the studies by Putta et al. [20] and Kalenda et al. [7] also reported the challenges and success factors of SAFe and LeSS. Three reviews reported the challenges and success factors of large-scale agile development in general instead of method-specific ([6], [22], [23]).

In summary, even though several reviews of the literature have been conducted in the related area, most of them are focused on agile transformation processes rather than large-scale methods per se. Those that do have a focus on large-scale methods are comparing methods primarily on the training or advertised materials of these methods and lack the required scientific empirical evidence. None of these reviews reported which principles, practices, tools, and metrics have been examined by the primary studies. Further, none of these reviews reported on custom-built (non-commercial) methods.

3 Research Process

We now present our strategy for the systematic review [46] and the threats to it's validity.

3.1 Search Strategy

Starting with the research questions, suitable keywords were identified using synonyms. We also extracted, aggregated, and used keywords from related literature reviews to build our search terms. The search terms were organised into three groups and separated by `AND-` clauses (see Appendix A). In the first group, we included methods that have been used in large-scale development. We used the latest report of VersionOne [47] to identify these methods. The second group ensured we retrieved papers associated with software and information systems development. The third group ensured that we included articles that discussed the use of agile methods in large-scale settings.

The digital libraries searched were Engineering Village, ISI Web of Science, Scopus and AIS e-Library to ensure coverage of computer science, software engineering (SE), and information systems (IS) literature [48], [49] (see Appendix A for the search string used in each digital library and the corresponding results). In total, we retrieved 35,215 articles.

3.2 Selection Strategy

The selection process is shown in Figure 1. Our search strings yielded broad search results, as we intended to err on the side of inclusivity by identifying as many papers as possible. This is typically done in a field such as this, where the area is not built on a cohesive body of well-defined terms [50]. Moreover, research on large-scale agile method is a cross-disciplinary study, and so terms used vary across disciplines.The first step in the selection process identified and removed a total of 33,150 irrelevant papers, duplicates, non-English, non-empirical papers, and secondary studies.

In the next step, we applied selection criteria (Table 1) to the 2,065 remaining papers. We employed the majority voting selection process [49]. Two independent reviewers evaluated each paper. For a paper to be included, two reviewers had to be in agreement. In cases where both reviewers did not agree, a third reviewer evaluated the record independently, and an agreement based on the majority was reached. At the end of this step, 1,700 papers were excluded.

The final step was to apply selection criteria to the full text of the 365 papers. This excluded a further 190 papers. During the full-text review, we also performed a snowballing search and found an additional 16 relevant articles. In total, 191 primary studies are included (see Appendix B for the complete list). Throughout the rest of the paper, each primary study is referred to by its identification number.

3.3 Quality Assessment

We performed a quality assessment on the 191 papers to evaluate the relative strength of empirical evidence or findings reported. The assessment (see Table 2) was based on the seven questions proposed by Dybå and Dingsøyr [51]. We used a three-point scale to answer the assessment questions: *no* (0), *to some extent* (0.5), and *yes* (1). The quality of each paper was assessed by two reviewers. Each reviewer could

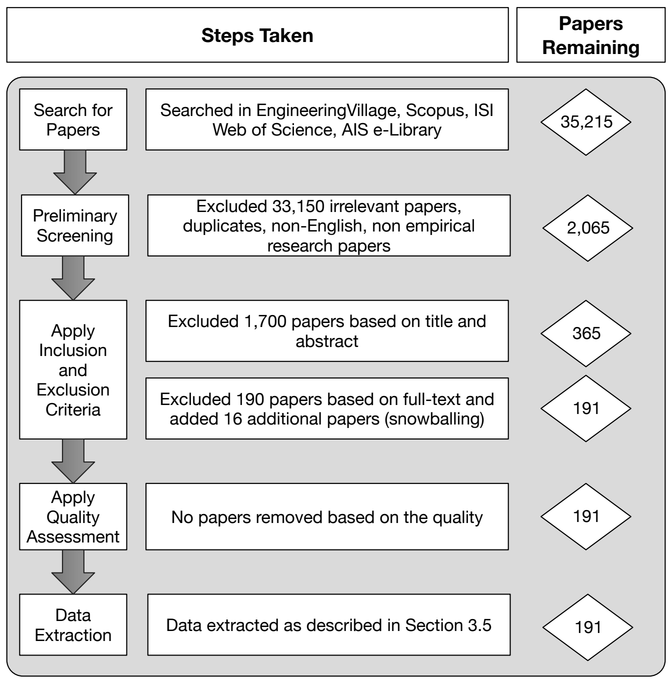
Steps Taken
Papers Remaining
Search for Papers
Searched in EngineeringVillage, Scopus, ISI Web of Science, AIS e-Library
35,215
Preliminary Screening
Excluded 33,150 irrelevant papers, duplicates, non-English, non empirical research papers
2,065
Apply Inclusion and Exclusion Criteria
Excluded 1,700 papers based on title and abstract
365
Excluded 190 papers based on full-text and added 16 additional papers (snowballing)
191
Apply Quality Assessment
No papers removed based on the quality
191
Data Extraction
Data extracted as described in Section 3.5
191

Figure 1: Steps for conducting the review

Table 1: Inclusion/Exclusion Criteria

Facet	Inclusion Criteria	Exclusion Criteria
Scale	(i) The method is practiced by at least 50 persons or six teams, working on a common product or project. (ii) The article explicitly considers itself as large-scale (either in title, abstract, or keywords).	(i) The process is practiced by a single team or less than 50 persons or six teams in a large or distributed setting. (ii) No indication that the context, process or issues are related to large-scale development. (iii) The method was not implemented in a real world development setting.
Context	Software or information systems development.	Non-software, e.g. manufacturing, supply chain, human resource, etc.
Studies	Peer-reviewed scientific papers, peer-reviewed or curated experience reports, written in English, full-text available.	Books, editorial, theses, position papers, talks, abstracts, non-English, and full-text not available.
Findings	The article provides at least one answer to our research questions.	The article does not provide an answer to the research questions.

give a maximum of 7 points to a paper; thus, the maximum possible score for the quality assessment for each paper was 14. Scores for primary studies ranged between 2 and 14. The median score was 8.0 and the average was 7.6. This indicates that the quality of the primary studies, on average, is good. The studies within the 25% percentile scored low on the quality assessment criteria regarding research design and data collection, data analysis, findings, and credibility of the conclusion. This is to be expected given that most of these studies are in the form of experience reports.

3.4 Data Analysis and Synthesis

We performed the data extraction during the full-text review. Each reviewer highlighted the relevant text in each article that corresponded to the properties listed in Table 3. The first reviewer aggregated the highlighted texts and compiled them into a spreadsheet for further analysis.

Table 2: Quality Assessment Checklist (adapted from [51])

Research Design
QA1. Is the research objective sufficiently explained and well-motivated?
QA2. Is the context (industry, project setting, product used, participants or observational units, etc.) in which the research was carried out clearly stated?

Data Collection
QA3. Was the data collection carried out thoroughly? For example, discussion of data collection procedures and how the research setting may have influenced the data collected.

Data Analysis
QA4. Are the approach to and formulation of the analysis well-conveyed? For example, justification and description of analysis method/tool/package.
QA5. Are alternative explanations and cofounders considered and discussed in the analysis?

Findings and Conclusion
QA6. Are the findings and conclusions credible? For example, the study is methodologically explained so that we can trust the findings; the findings are clearly stated and supported by the data, and are resonant with other knowledge and experience.
QA7. Are limitations and credibility of the study adequately discussed?

Table 3: Extracted Data

	Property	Goal
D1	Research Method	Overview
D2	Context (Organisation name, size, development period)	Overview
D3	Large-scale method, connecting practices	RQ1
D4	Principles, practices, tools, metrics	RQ2
D5	Challenges	RQ3
D6	Success factors	RQ4

Data for D1 and D2 were collected to provide a broad overview of the primary studies. We categorised the primary studies according to the applied research methods. When a study reported industrial experiences without stating research questions or research methods, it was classified as an experience report.

We recorded data for D3 to D6 to answer our research questions. To help answer RQ2, we searched for the official sources of each large-scale development framework in either published books or websites and identified the prescribed principles, practices, tools, and metrics. We searched for the most recent main sources of SAFe[2], LeSS[3], Scrum-at-Scale[4], DAD[5] and Spotify model[6]. Then, we examined each primary study to evaluate whether it discussed or reflected the findings at each abstraction level. Using the definitions given in Section 1.2, we identified the principles/practices/tools/metrics studied in these primary studies and mapped them to the ones in the official sources.

2. https://www.scaledagileframework.com/
3. https://less.works
4. https://www.scrumatscale.com/
5. https://disciplinedagiledelivery.com/
6. https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf;https://vimeo.com/85490944;https://vimeo.com/94950270

3.5 Threats to Validity

This subsection describes four potential validity threats to this systematic review. In addition we also discuss our strategy to address each.

3.5.1 Publication Bias

Publication bias occurs when positive research outcomes are more likely to get published than negative ones [46], [52]. Method creators often have a vested interest in the reporting of the method, and organisations studied may wish to avoid or at least reduce the emphasis on negative aspects of them. Review studies such as ours are always susceptible to such bias. In this study, we regard this threat as high. To mitigate it, we conducted a quality screening check, and distinguished between rigorous empirical research and experience reports. We decided not to include grey literature, e.g. work-in-progress, technical reports, blogs, and unpublished or non-peer reviewed articles.

3.5.2 Threats Regarding Identification of Primary Studies

To reduce the threat of missing or excluding relevant papers, our strategy was to retrieve as many peer-reviewed papers as possible related to large-scale software or information systems development. We also performed a snowballing search during the full-text review by finding relevant papers in the reference list of each paper [49]. It is impossible to completely eliminate the threat of missing relevant articles. Inconsistent terminology, particularly in large-scale software development research, or use of different terminology with respect to the exercised search string (Appendix A) may have biased the identification of primary studies.

We did not attempt to optimise the search string for a high level of precision, in order to be as inclusive as possible. Precision refers to the ratio of retrieved relevant items and all retrieved items [53]. Our precision is 1%, considering 22,967 unique papers (without duplicates) and 191 primary studies. The low precision causes more effort to select the primary studies, as discussed in Section 3.5.3.

3.5.3 Threats to Consistency of Selection of Primary Studies and Data Extraction

As discussed in Section 3.2, the formulation of the inclusion and exclusion criteria involved a pilot at each stage to remove ambiguities and to check the agreement level among reviewers. The first pilot was based on the title and abstract. The assessment results of each reviewer were compared. A meeting was called to discuss the contrasting interpretation of the criteria and disparities in the assessment results. A similar approach was followed during the second pilot on the full-text review. An ambiguity revealed during the piloting was related to the definition of *large-scale*, as some studies did not explicitly report the size of the case under the study. Definitions were thus formally made and agreed upon. The pilot exercise was re-run, and it was clear that this issue had been adequately resolved.

A pilot was also conducted on the quality assessment checklist. Similar to the selection process, two independent reviewers assessed the quality of the primary studies. A pilot with 50 papers was performed to check the agreement level among the reviewers. After the pilot, a meeting was called to evaluate the checklist and find strategies to approach certain vagueness or missing details in some papers. Any dissimilarity in the reviewers' assessment was also discussed and clarified in the presence of all reviewers.

3.5.4 Reliability of the Extracted Data

We found that some studies did not have clear details about the team size or about the large-scale method. In this case, we had to infer using researcher judgment. For example, some papers did not reveal the number of development teams but rather referred to the whole organisation. In the cases where the method was not clearly stated, we used the study context and practices reported in all related studies to classify the paper. Therefore, there is a possibility that some of the extracted findings are partially inaccurate. To mitigate this, while performing a full-text review, each reviewer refined and verified the extracted data. The extracted data were then re-checked by the other reviewers.

4 Characteristics of Primary Studies

4.1 Publication Types, Sources, and Venues

The distribution of primary studies by year is shown in Figure 2, which also highlights the year in which some of the well-known large-scale agile frameworks were proposed. All primary studies were dated after the year 2001. There were peaks in 2007 and 2013, and the number of studies increases steadily since 2016 till the time of this study. The empirical studies of a particular large-scale agile framework were typically published three years after its adoption in the organisation under the study.

The majority of primary studies (83%) are published in SE-related conferences, e.g. Agile Conference, XP, International Conference on Global Software Engineering, or journals, e.g. Empirical Software Engineering, and IEEE Transactions on Software Engineering. The topic is also of interest to IS researchers as 11% of the primary studies are published in IS conferences, e.g. Hawaii International Conference on System Sciences, Americas Conference on Information Systems, or journals, e.g. Communications of the Association for Information Systems or MIT Sloan Management Review. In Tables 4, 7, 6, 8, and 9, a double asterisk sign (**) indicates studies published in IS discipline. We also identified studies published in other disciplines (6%) such as management (e.g. Project Management Journal, International Journal of Managing Projects in Business), social science (e.g. Industrial and Corporate Change) or design (e.g. International Design Conference).

Most of the 191 primary studies are empirical research papers (66%), whilst the rest are experience reports (34%). In Tables 4, 7, 6, 8, and 9, a single asterisk sign (*) indicates experience reports. Among the 127 empirical research papers, only 10 (8%) used theories. For the purpose of this work, we adopted Gregor's [54] understanding of theory: (i) generalisation – an attempt to generalise knowledge of specific events or object into more abstract and universal, (ii) causality – the relationship between cause and effect, and (iii) explanation and prediction – understanding and predicting a phenomenon and guiding action. In the reviewed empirical studies, theories were often used to explain how and why some phenomena occurred, e.g. coordination mode

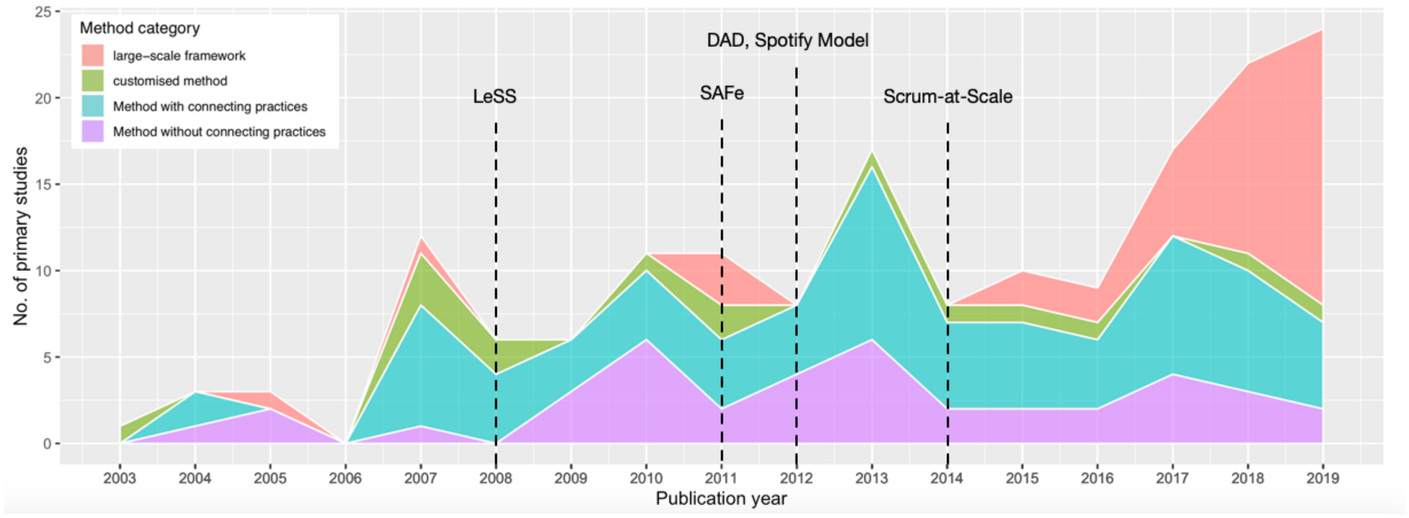

Figure 2: Temporal distribution of the primary studies by method category

and mechanism theory (PS23, PS38, PS88), trust–mediated organisational control (PS31). Theories were also used to understand the causality between people and technology, e.g. socio-technical systems theory (PS26), project governance (PS71), knowledge management theory (PS109), relational coordination theory (PS146), routine dynamics theory (PS152), and adaptive structuration theory (PS180). There is no predominant theory used in the reviewed studies.

4.2 Authors, Case Organisations, and Primary Studies

The 191 primary studies are written by a total of 325 authors. On average, there are four authors per article, in which one of them had an affiliation with the case organisation when the paper was published. As shown in Figure 3, the majority of the authors come from Scandinavia (37%) then followed by North America including the United States and Canada (20%). Figure 4 shows the distribution of the case organisations across business sectors. Twenty-six primary studies (19%) were conducted in software vendors, while sixteen were in telecom companies, e.g. Ericsson or Nokia.

In total, there were 134 case organisations studied. In several cases, more than one primary studies focused on the same organisation, but usually focused on different aspects. For example, one paper may study inter-team coordination (PS11), while others would examine retrospectives (PS21) or customer involvement (PS22). Conversely, if a paper studied multiple organisations, each organisation is treated as an individual case, e.g. PS124, PS132. However, if those organisations were working on the same project, they are all treated as one case e.g. PS135. Some organisations may have branches, subsidiaries, or business units in multiple locations and work on different products, e.g Siemens Healthineers (PS36, PS37, PS84, PS111, PS149, PS153, PS174) or Siemens Technology and Services (PS113, PS127, PS128, PS179). We treated them as separate case organisations. The list of case organisations is presented in Appendix C. Similar to the primary studies, the unique case organisation is referred to using their case IDs.

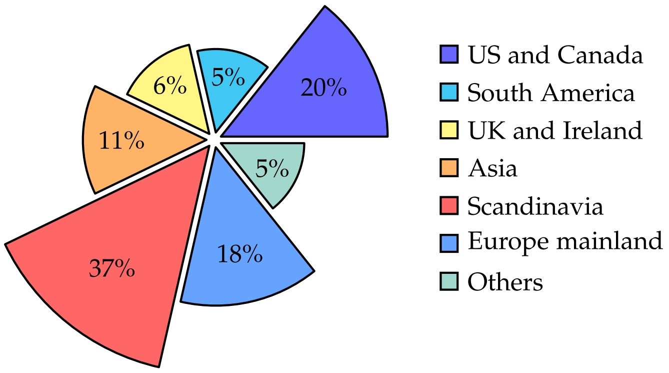

Figure 3: Distribution of the authors' countries

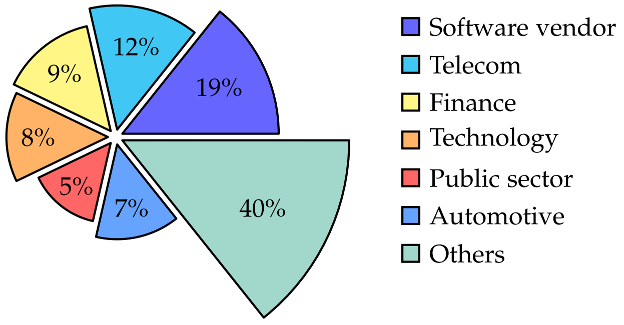

Figure 4: Business sector distribution of the case organisations

5 Results and Analysis

This section provides an overview of large-scale agile methods based on a synthesis of the reviewed primary studies[7].

5.1 Large-Scale Agile Methods

Table 4 presents the methods/frameworks, case organisations and the primary studies investigating these methods

7. Seventeen primary studies (PS3, PS4, PS5, PS44, PS53, PS133, PS141, PS143, PS145, PS154, PS155, PS156, PS168, PS175, PS176, PS180, PS183) were not used in the analysis as they studied various methods at multiple organisations and reported the results in an aggregated manner. Thus it was impossible to extract method-specific information.

in the case organisations. Two articles (PS82, PS89) appear in the table multiple times as they reported multiple case organisations that used different methods. Similarly, some case organisations (e.g., CO50, CO53) appear in the table multiple times as they may have adopted different large-agile agile methods at different points of time.

As shown in Table 4, the predominant method used in large-scale development is Scrum (49 organisations), either as a single method or combined with other methods, followed by SAFe (19 organisations).

Forty-one primary studies reported the use of commercial large-scale methods in 32 organisations. SAFe alone was used in 19 organisations (21 primary studies), while one study reported the use of both SAFe and Spotify model in a single case (CO129, PS171). LeSS was used in four organisations (11 primary studies). Some organisations were using a traditional, plan-driven method before deciding to adopt a large-scale agile framework. Some were already familiar with agile methods prior to the large-scale framework adoption. These frameworks aim at addressing large-scale development issues, such as co-ordination of multiple teams, and requirement analysis (PS91, PS104).

Fifteen primary studies reported that 13 organisations did not adhere to a specific large-scale method, but combined and tailored multiple methods for large-scale development to suit their particular context. For example, four frameworks were based on Scrum: Distributed Scrum (PS17), Adaptive Development Methodology (ADM - PS20, PS67, PS80), Product Evolution Process (PEP - PS179), and Enterprise Scrum (PS32, PS33). Two frameworks were based on the Unified Process: BEKK Model (PS2), PAF Model for Project (PAMP - PS103), and one was based on XP: Qwest's Enterprise XP (PS121). All of these customised frameworks were typically based on the processes that the case organisations were already familiar with, which were previously implemented at a smaller scale. As the numbers of teams and projects grew, management needed to formalise and control the processes across the organisation. They then selected, tailored, and integrated different existing method fragments relevant to their context. The customised frameworks were typically implemented only in one case organisation only, except HAMRA, which was used in two case companies (CO33, CO34) as reported in PS30.

Among the 113 primary studies that did not mention the use of commercial large-scale frameworks, we found that 72 studies reported connecting practices. Literature shows that large-scale development involves challenges related to inter-team coordination, large project organisation, release planning and architecture, customer collaboration (including contracts), and knowledge sharing and improvement [55]. Thus, when a particular method is being scaled in large-scale development, we expect to see connecting practices to address these challenges. Table 5 tabulates connecting practices reported in these primary studies. For improving inter-team coordination, the Scrum-of-Scrums (SoS) meeting is the most common connecting practice reported in 17 primary studies of 15 case organisations), in which Scrum masters from each team meet to coordinate the delivery of software and solve inter-dependencies. Four primary studies reported that four organisations established a steering committee, council or project management office at the portfolio level to make decisions regarding the overall project and process improvement. Another connecting practice suggested architecture guidelines be used for development teams. This was implemented in ten organisations and reported in nine primary studies. To leverage customer collaboration, scaling the role of the Product Owner (PO) was a common practice in five organisations (12 primary studies). The common practices to improve knowledge sharing are community of practice (seven primary studies) and shared resources e.g. web-based wiki (eight primary studies). In contrast, we did not find any evidence of connecting practices reported in 41 studies.

Figure 2 illustrates the evolution of the primary studies by the four categories of large-scale agile software development methods. It is not surprising to see the uptake of studies focused on large-scale agile frameworks, especially in recent years. However, it is also noticeable that there is a stream of studies investigating large-scale development in the organisations that did not apply any large-scale methods (highlighted in light-blue and purple). The number of these types of studies did not diminish despite the emergent and prevalence of large-scale frameworks such as SAFe. In comparison, the studies of customised large-scale methods (highlighted in green) are marginal.

5.2 Levels of Abstraction of Large-scale Agile Frameworks

Table 6 lists the studies that investigated the large-scale agile frameworks (the first group in Table 4) at various levels of abstraction (principles, practices, tools, and metrics). Table 7 lists these elements in each framework from the official sources and the emergent ones from the primary studies.

5.2.1 *Principles*

Each large-scale framework studied included basic principles. For example, SAFe explicitly mentions four core values (alignment, built-in quality, transparency and program execution), Lean-Agile mindset and ten principles. LeSS has ten principles. DAD has seven principles, while Scrum-at-Scale defines five core values (openness, courage, focus, respect, and commitment) and Spotify model has ten. In this study, we grouped the terms *value*, *mindset*, and *principle* together under *principle*, as they inspire and inform the practices, tools and metrics of a framework. From this point on only the term *principle* is used.

Our review found that only two out of 21 SAFe-related studies focused on a lean-agile mindset (PS24, PS101). SAFe is built on the agile values and principles, and methods. Thus the adoption of SAFe requires full support from leaders to embrace agility across an organisation. SAFe does not focus on team process or workflow, but rather on the adaptation of processes, roles, and tools parallel in the whole organisation (PS24, PS101).

The study PS139 reported two emerging principles of the Spotify model: *building structure around customer needs and keep it fluid*, and *providing employees with development and growth opportunities*. The study PS163 suggested four principles of a tailored Spotify model, but this was specifically in business-to-business (B2B) organisations: *strengthening the interactions within B2B product development, building strong*

Table 4: Large-Scale Development Methods/Framework, Organisations and Primary Studies

Method/Framework		Case Organisation	Source(s)	Frequency	
Category	Primary method/framework			Case	Source
Large-scale development frameworks	SAFe	CO13, CO14, CO38, CO40, CO41, CO42, CO53, CO60, CO61, CO88, CO91, CO95, CO101, CO118, CO119, CO120, CO122, CO123, CO127	PS24, PS91, PS35*, PS38**, PS66, PS68*, PS101, PS104, PS105, PS122, PS142, PS144, PS148*, PS150, PS151, PS152**, PS157, PS161*, PS162, PS167, PS178**	19	21
	LeSS	CO6, CO50, CO105, CO128	PS10, PS11, PS21, PS22, PS23, PS83, PS88, PS97, PS125, PS147*, PS170	4	11
	DAD	CO7, CO62	PS14*, PS70	2	2
	Spotify Model	CO116, CO124, CO126	PS139*,**, PS163, PS164, PS166	3	4
	Scrum-at-Scale	CO102, CO110, CO111	PS123, PS132*,**	3	2
	SAFe & Spotify Model	CO129	PS171	1	1
Customised large-scale frameworks	HAMRA	CO33, CO34	PS30	2	1
	ADM	CO12	PS20*, PS67, PS80	1	3
	Enterprise Scrum	CO36	PS32*,**, PS33*,**	1	2
	BEKK Model	CO2	PS2*	1	1
	Promotion Process	CO4	PS51*	1	1
	Lean Change	CO48	PS46*	1	1
	ASSF	CO89	PS102	1	1
	PAMP	CO90	PS103**	1	1
	Distributed Scrum	CO9	PS17**	1	1
	Qwest's Enterprise XP	CO100	PS121*	1	1
	Flow	CO40	PS153*	1	1
	PEP	CO96	PS179*	1	1
Methods *with* connecting practices	Scrum	CO3, CO4, CO5, CO10, CO15, CO16, CO17, CO18, CO19, CO20, CO35, CO40, CO43, CO45, CO47, CO49, CO50, CO51, CO52, CO53, CO54, CO58, CO63, CO64, CO65, CO69, CO83, CO84, CO95, CO109, CO117, CO121, CO132	PS6*, PS7, PS8, PS18*, PS69, PS82, PS117, PS131*,**, PS25, PS31**, PS43, PS41, PS45*, PS47, PS49, PS50*, PS54, PS55*, PS60*, PS71, PS72, PS73, PS75*, PS136*, PS84*, PS93, PS96, PS112*, PS130*, PS110, PS140*, PS146, PS177, PS188*, PS189	34	35
	Lean	CO5, CO43, CO74, CO87	PS9, PS81, PS100*, PS107	4	4
	XP	CO8, CO131	PS16**, PS187*	2	2
	Kanban	CO55, CO95, CO103, CO104	PS56, PS119*, PS124	4	3
	Scrum & XP	CO10, CO37, CO85	PS34*, PS98*, PS181*	3	3
	Scrum & SPLE	CO31, CO39, CO44, CO71, CO72, CO73	PS27, PS40, PS78, PS79, PS174*	6	5
	Scrum & Kanban	CO5, CO77	PS87, PS184	2	2
	Lean & Scrum	CO4, CO39, CO43, CO130	PS12, PS115*, PS36*, PS37*, PS111*, PS149*, PS39*, PS90, PS92, PS94, PS95, PS159, PS173*, PS185, PS186	4	15
	Continuous Delivery (CD)	CO75	PS85*	1	1
	Continuous Integration (CI)	CO65	PS77	1	1
	DevOps & Scrum	CO133	PS190**	1	1
Methods *without* connecting practices	Scrum	CO1, CO4, CO46, CO57, CO92, CO94, CO97, CO106, CO107, CO108, CO112, CO113, CO114, CO115, CO134	PS1*, PS114, PS42, PS59*, PS106*, PS109, PS116*, PS126, PS129**, PS134**, PS135**, PS137*, PS138, PS191	15	14
	Lean	CO43, CO50, CO59, CO86, CO96	PS52, PS48, PS65, PS99*, PS113*, PS127*, PS12*	5	7
	XP	CO50, CO66, CO67, CO68, CO76	PS74, PS86*	5	2
	Kanban	CO70	PS76*	1	1
	DevOps	CO21, CO22, CO23, CO24, CO25, CO26, CO27, CO28, CO29, CO30, CO99	PS26**, PS120*	11	2
	Scrum & XP	CO32, CO51	PS28, PS29, PS61, PS62, PS63, PS64	2	6
	Scrum & Lean	CO56	PS58	1	1
	V-Model	CO66	PS82	1	1
	CD	CO98	PS118*	1	1
	Agile R&D	CO78	PS89	1	1
	CI	CO79, CO80, CO81, CO93	PS86*, PS105	4	2
	Continuous Deployment	CO11, CO82	PS19, PS89	2	2
	Lean Startup	CO125	PS165	1	1
Not stated			PS13, PS15, PS57, PS158, PS160, PS169, PS172, PS182	0	8

*Note: * indicates experience reports, ** indicates articles published in IS venues.*

and successful B2B relationships, strengthening project visibility for customers, and satisfying customers by responding at different velocity levels. We did not find any primary study that examined the principles of LeSS, Scrum-at-Scale or DAD.

5.2.2 *Practices*

In this category, we grouped all practices and related terms, such as structures and artefacts provided by a large-scale framework. For example, SAFe provides eight practices,

Table 5: Connecting Practices

Areas	Practices	Case ID	Source(s)
Inter-team coordination	SoS meetings (i.e. Grande SoS, feature SoS, etc.)	CO10, CO37, CO39, CO43, CO52, CO54, CO58, CO40, CO77, CO83, CO84, CO85, CO130, CO132, CO133	PS34, PS36, PS37, PS50, PS55, PS60, PS84, PS87, PS93, PS96, PS98, PS111, PS173, PS181, PS186, PS189, PS190
	Central team directives	CO4, CO39, CO43, CO69	PS7, PS8, PS75, PS90, PS111, PS186
	Joint/inter-team retrospective	CO69, CO77, CO87, CO117	PS75, PS87, PS100, PS110
	Iterative proxy collaboration	CO4, CO10, CO43	PS7, PS8, PS18, PS95, PS107, PS131
	Visualisation (i.e. dependencies, deliveries, IT project portfolio)	CO65, CO74, CO103, CO104	PS73, PS87, PS81, PS124
	Common goal for the sprint	CO69, CO109	PS130, PS136
	Team specific swim lane	CO103, CO104	PS124
	Central team planning	CO4	PS7, PS8
	Ad-hoc communication	CO4	PS7, PS8
	Synchronised sprint cycle	CO64, CO65	PS72, PS77
	Virtual stand up meetings	CO39	PS37
	Mid sprint review	CO42	PS38, PS152
	Theme review meetings	CO69	PS75, PS136
	Collaborative platform	CO55	PS56
	Cross-team demo	CO43, CO69	PS75, PS136, PS186
	PO coordination meetings	CO43, CO124	PS146, PS186
Large project organisation	Steering committee, portfolio council or project management office	CO3, CO47, CO95, CO112	PS6, PS45, PS119, PS134
	Establishing central roles	CO5, CO69, CO83, CO84	PS75, PS96, PS184
	Establish a proxy between management and development	CO43, CO95	PS90, PS112
	Flat organisational hierarchy	CO8	PS16
Release planning and architecture	Architecture guidelines	CO31, CO35, CO39, CO44, CO47, CO49, CO71, CO72, CO73, CO40	PS27, PS31, PS40, PS45, PS47, PS78, PS79, PS84, PS174
	Regular full integration of software, hardware, and mechanics	CO15, CO16, CO17, CO18, CO19, CO20	PS25
	Strategic roadmap / high level planning	CO43, CO47, CO65, CO69, CO75, CO94, CO117	PS45, PS73, PS75, PS85, PS90, PS109, PS110
	Joint release planning	CO4, CO45	PS7, PS8, PS41
	Requirements architect role	CO3, CO39	PS6, PS111
Customer collaboration	Scaling the POs (e.g. one PO in each team)	CO43, CO51, CO69, CO83, CO109	PS43, PS49, PS75, PS92, PS93, PS94, PS95, PS96, PS107, PS130, PS136, PS186
	Network of POs	CO39	PS111
Knowledge sharing and improvement	Training, workshops and seminars	CO35, CO39, CO43, CO58, CO95	PS31, PS60, PS92, PS111, PS112
	Community of Practice (CoP)	CO5, CO43, CO109, CO117	PS82, PS92, PS94, PS95, PS107, PS110, PS130
	Team member rotation	CO37, CO65, CO74	PS34, PS73, PS81
	Physical proximity of teams	CO5, CO43, CO133	PS39, PS69, PS107, PS117, PS190
	Shared workspace and source	CO10, CO43, CO131	PS43, PS90, PS92, PS94, PS95, PS107, PS181, PS187
	Flow-assisted Value Stream Mapping	CO5	PS9
	Posting progress for a constant feedback	CO74	PS81
	Technical experts as a point of contact	CO5	PS82, PS184

LeSS has ten Scrum-based practices and five structures. In the Spotify model, teams are organised into squads, tribes, guilds and chapters, and adopt Lean startup practices such as *minimum viable product* and *validated learning*. Similarly, Scrum-at-Scale prescribes practices based on Scrum, while in DAD practitioners can choose practices based on Scrum, Lean, Continuous Delivery life cycle.

Eighteen out of 21 SAFe studies focused on practices. For example, *Program Increment (PI) planning* and *Agile Release Train (ART)* are the two practices studied by the majority of the studies (e.g. PS24, PS35, PS38, PS91, PS101, PS148, PS150, PS151, PS152, PS162, PS167, PS178). Only one study examined the SAFe portfolio management practices (PS68). PS148 suggested a new role *Complete System Architecture (CSA)* to solve dependencies between multiple ART and/or solutions. In the LeSS studies, various practices for coordination and integration have been studied empirically, e.g. establishing various types of meetings overtime, continuous integration, Community of Practices (PS10, PS22, PS23, PS83, PS88, PS97). Moreover, two studies (PS11, PS12) specifically reported various team and organisational structures to support LeSS. PS11 suggested a new practice *mini-demos* to improve the communication between the developing and business teams. Instead of having a common demo at the end of every iteration, mini-demos can be held during iterations to receive feedback from the business team.

PS163 and PS164 analysed several practices of the Spotify Model in a mission-critical software service, including fail-friendly environment, features with a toggle switch, embrace Lean startup to promote innovation, and suggested software product lines for automation and standardisation. PS139 suggested two new practices for the model, *pop-up squads* to manage one-off, short-term projects, and *quarterly business review (QBR) meetings* where each tribe lead defines a set of objectives and key results for the following quarter.

PS123 and PS132 reported the scaling of SoSoS (Scrum-of-Scrum-of-Scrums) in Scrum-at-Scale. A SoS (Scrum-of-Scrum) operates as a Scrum team with scaled versions of

Table 6: Levels of large-scale method's abstraction studied by primary studies

Category	SAFe		LeSS		Scrum-at-Scale		DAD		Spotify	
	Sources	Frequency	Sources	Frequency	Sources	Frequency	Sources	Frequency	Sources	Frequency
Principles	PS24, PS101	2/21	-	0/11	-	0/2	-	0/2	PS139*,**, PS163	2/4
Practices	PS24, PS35*, PS38**, PS66, PS68*, PS91, PS101, PS122, PS142, PS144, PS148, PS150, PS151, PS152**, PS157, PS162, PS167, PS178**	18/21	P10, PS11, PS21, PS22, PS23, PS83, PS88, PS97, PS125, PS147*, PS170	11/11	PS123, PS132*,**	2/2	PS70	1/2	PS139*,**, PS163, PS164, PS166	4/4
Tools	-	0/21	-	0/11	-	0/2	-	0/2	-	0/4
Metrics	PS104, PS105, PS161*	3/21	-	0/11	-	0/2	PS14*	1/2	-	0/4

*Note: * indicates experience reports, ** indicates articles published in IS venues. A cell with gray background indicates that there are no elements at that level of abstraction prescribed by the framework.*

roles, events, and artefacts. For example, for each SoS in a large distributed team, there is a group of PO (at program and team levels) or Scrum Masters (PS123, PS132). Only one study investigated the practices of DAD (PS70).

5.2.3 Tools

We grouped all supporting tools prescribed by the frameworks (see Table 7). SAFe provides various tools for different purposes, e.g. feature progress chart, program Kanban boards, burn-up charts, and continuous flow diagrams to increase the visibility and transparency of program performance. Scrum-at-Scale does not explicitly describe any tools to support its usage. Correspondingly, we did not find any related studies. Surprisingly, neither did we find any studies focusing on the tools of the other frameworks.

5.2.4 Metrics

SAFe provides 20 types of metrics for all levels of the framework, e.g. business agility self-assessment for the enterprise level, value stream key performance indicator or lean portfolio metrics for the portfolio level. We found three dedicated studies on the metrics of SAFe (PS104, PS105, PS161). The study PS104 and PS105 examined two self-assessment measurements provided by SAFe: Lean enterprise assessment and Lean portfolio metrics. The studies argued that these metrics were useful to assess how well the adoption of SAFe practices in an organisation was. PS161 proposed SAFe-based metrics to measure the quality in a hybrid development model.

One study of DAD (PS14) focused on metrics used to measure improvements from two perspectives: business-related and agile-related. Business-related metrics were used by leaders and executives, while agile-related metrics were used by the teams. Our review did not find any studies on the metrics of LeSS, Scrum-at-Scale, DAD or the Spotify model, in accordance with the fact that no metrics are specified in these frameworks (except for Scrum-at-Scale).

5.3 Challenges of Applying Large-Scale Agile Methods

This research question aimed to identify the challenges that organisations are confronted with when applying large-scale agile development methods. We organised the 31 challenges that were reported in the primary studies into nine categories, as shown in Table 8.

5.3.1 Inter-team Collaboration

Synchronising across dynamic and fast-moving teams (C-IC-1) has been a challenge for organisations that adopted Scrum-at-Scale (PS132), custom built-methods (PS17, PS166, PS179), or scaled methods (PS36, PS149). For example, in the case of CO9, CO110, and CO111, the challenge with team synchronising is because each team is responsible for different tasks, thus it is important to assure the know how transfer and the update of the deliverable to the next team and all stakeholders (PS189). Moreover, the challenge is also to synchronise tasks to reduce dependency while at the same time to maintain consistent performance across the dynamic and fast-moving teams (PS17, PS132).

SoS meetings may not be an effective communication channel to discuss and address impediments (PS36). One reason is due to the multiple agile layers (C-IC-2). As the number of layers grows larger, meetings take longer to cover all topics and teams may not receive any feedback or solutions to their problems (PS49, PS96). The attendance in other common meetings, e.g. common sprint planning, common demo, common retrospective was seen as waste (PS97, PS186). This is also the case for SAFe. Spending too much time in joint meetings, e.g. Product Increment (PI) Planning meetings, was considered waste (PS150, PS157).

The study PS84 reported that the development Scrum teams of CO40 struggled to avoid distractions that prevented them from concentrating on development tasks (C-IC-3). For example, team members must participate in cross-team activities and allocate time to project-wide activities, e.g., Scrum meetings or mentoring:

> "While [team members] recognized the need for cross-team communication, they resented context switching and the impact on their time dedicated to the team's spirit goals." (Scrum, CO40, PS84)

While scaling the Lean and Scrum methods to large-scale development, enabling end-to-end development (C-IC-4) and maintaining transparency across a high number of fast-moving, adaptive teams and projects (C-IC-5) remained challenging for the development teams of Ericsson (CO43). All needed knowledge and infrastructure should be available for each site to enable end-to-end development (PS95). Moreover, some teams limited the collaboration within the agile teams. This is risky as it can lead to local optimisations at the overall systems' expense (PS107). Moreover, achieving end-to-end flow may increase handovers (considered a type of waste by Lean) as it involves a number of decision-making processes (PS107).

5.3.2 *Organisational Structural Challenges*

Agile principles require organisations to have cross-functional and fast-moving teams that can implement any features of software (PS25, PS43, PS49, PS92). However, it may be the case that some features are very domain-specific and require additional competency and expertise before development starts. Therefore, management assigned features to specialist teams that had the required competency. Balancing between building generalist and specialist teams (C-OS-1) is even more challenging as the pressure to release the features is mounting (PS43, PS92).

In some cases, when organisations attempt to scale a particular agile method for large-scale development, the definitions of new roles and responsibilities (C-OS-2) are not always straightforward (PS46, PS49, PS92). This typically happens in an organisation where agile teams are created in a traditional waterfall organisation and the agile roles are practised in traditional manner (PS49). The inadequate transition of organisational roles could lead to (i) delegation instead of self-organisation, (ii) unbalanced workload, (iii) divided business and organisation, and (iv) unproductive agile ceremonies (PS49).

The emergence of various new roles while adopting SAFe could lead to a complex organisational setup (C-OS-3, PS101). It causes a large number of handovers (PS142). Information to and from development teams must go through multiple levels in the organisation, which are considered to affect the development speed negatively (PS101, PS142).

A race condition might happen in an organisation as multiple teams compete for shared resources (C-OS-4) to meet the planned schedules (PS33). For example, quarterly sprints of Business Scrum suggest that several releases could occur at the end of a quarter, but at the same time other teams such as testing, customer support also need to deliver value to customers. These teams may experience fluctuating demand and need to prioritise their work.

5.3.3 *Architectural Challenges*

The inability to see the big picture at the program level (C-AR-1) is perceived as a challenge when applying SAFe (PS24), LeSS (PS22), custom-built framework (PS17), and scaled methods (PS36, PS55, PS117, PS124, PS140, PS149). For example, one team is responsible for completing a feature request while another team takes care of test cases. When the work needs to be passed from one team to the other, it is difficult to keep the flow as they cannot describe the overall system's behaviour (PS17). One reason is no support structure in the organisation above the team level:

> "Although [SAFe] development worked relatively well at the team level, the team members had difficulty seeing how their daily work linked to and affected other parts of the globally distributed organization." (SAFe, CO13, PS24)

The study PS36 also shows that ineffective SoS meetings challenged teams in presenting the big picture (e.g. product performance of previous releases, major customer pain areas, technical debts, overall project status, major feedback from specialist) to the team members and stakeholders (C-AR-1). The transparency among stakeholders becomes even lower with an increasing number of teams scaling up and being further distributed geographically (PS36, PS124).

Missing the big picture of the systems may affect the continuous integration (CI) and test automation process (C-AR-2, PS117). Teams may also receive incompatible components developed by other teams, which can result in merge conflicts. Often, this causes teams to re-work and ultimately leads to late deliveries or releases (PS117). While a strong CI pipeline is considered a major facilitator for SAFe, building CI and test automation requires a huge effort (PS92).

PS132 and PS191 reported the challenges related to agile and security. As large-scale projects involve multiple agile teams, organisations need to ensure confidential electronic transmissions and privacy (PS132). It could be more challenging when a project involves external developers who focus on their assignments mainly and lack security awareness (C-AR-3, PS191). In the case of CO134, security testing activities took place only if an item for a certain security feature is present in the backlog. Other security-enhancing activities commonly recommended for agile projects such as security code reviews were not performed.

5.3.4 *Requirements Engineering Challenges*

The importance of requirement planning across agile teams was pointed out in PS117, as it would minimise the occurrence of dependencies. However, in a large-scale project where managers lack knowledge on software development, it might happen that plans are not developed appropriately. Insufficient planning leads to dependencies across teams. Ultimately, it will lead to changes in requirements and time management plans (C-RE-1, PS25, PS41, PS54, PS55, PS117).

Capturing value and prioritisation (C-RE-2) is considered challenging in scaled methods (PS46, PS93, PS117). Unlike in small and medium projects, identifying and prioritising the right requirements should consider all stakeholders involved in a large-scale project. Conflict may arise as different customers and teams have different values and prioritisation. The responsibility for capturing and maximising product value should be shared among stakeholders.

To be able to deliver working software frequently, agile suggests breaking down requirements into smaller user stories, which can be done in a short timescale. However, it needs more effort to maintain and trace them (C-RE-3). Moreover, formulating user stories that are clear, detailed enough, and measurable for both product and non-product projects is difficult (PS130). Organisations, especially at the team level, struggle to break down the requirements into smaller user stories to implement them within one release. Even though this seems to be a minor issue, it does cause many serious problems (PS92). If user stories are too big, it is difficult to see the progress of a development team (PS124):

Table 7: Principles, practices, tools and metrics of main large-scale methods

	Official Sources	Emerging Elements from Primary Studies
	SAFe	
Principles	*Core Values:* Alignment, Built-in quality, Transparency, Program execution *Lean-Agile Mindset:* SAFe House of Lean and Agile Manifesto *Principles:* (i) Take an economic view, (ii)Apply systems thinking, (iii) Assume variability; preserve options, (iv) Build incremental with fast, integrated learning cycles, (v) Base milestone on objective evaluation of working systems, (vi) Visualise and limit WIP, reduce batch sizes, and manage queue length, (vii) Apply cadence; synchronise with cross-domain planning, (viii) Unlock the intrinsic motivation of knowledge workers, (ix) Decentralise decision-making, (x) Organise around value	
Practices	Build solutions components with high functioning ARTs, Build and integrate with a solution train, Capture and refine systems specification in solution intent, Apply multiply planning horizons, Architect for scale, modularity, reusability, and serviceability, Manage the supply chain with 'systems of systems' thinking, Apply 'continuous integration', Continually address compliance concerns	Complete System Architecture (CSA) role (PS148*), Mid sprint review (PS38**)
Tools	Progress feature chart, program Kanban boards, burn-up charts, continuous flow diagram	
Metrics	Lean Enterprise Assessment, Lean Portfolio Metrics, Enterprise Balanced Scorecard, Lean Portfolio Management Self-Assessment, Value Stream Key Performance Indicators, Feature Progress Report, Program Predictability Measure, Performance Metrics, Cumulative Flow Diagram, Agile Release Train Self-Assessment, Continuous Delivery Pipeline Efficiency, Deployments and Releases per Timebox, Recovery over Time, Innovation Accounting and Leading Indicator, DevOps Health Radar, Solution Predictability Measure, Solution Performance Metrics, Iteration Metrics, Team PI Performance Report, SAFe Team Self-Assessment	SAFe-based metrics for hybrid development model (PS161*)
	LeSS	
Principles	(i) Large-Scale Scrum is Scrum, (ii) More with LeSS, (iii) Lean Thinking, (iv) Systems Thinking, (v) Empirical Process Control, (vi) Transparency, (vii) Continuous Improvement Towards Perfection, (viii) Customer Centric, (ix) Whole Product Focus, (x) Queueing Theory	
Practices	Sprint planning, Sprint review, Retrospective, Overall retrospective, Daily Scrum, Coordination and Integration (Just talk), Communicate in code (continuous integration), Send observer to the Daily Scrum, Component communities and mentors (e.g. CoPs), Scrum of Scrums, Multi-team meetings, Travellers to exploit and break bottlenecks and create skill, Leading team), Requirement Areas, Area Product Backlog, Area Product Owner Technical Excellence (Specification by Example Continuous Delivery, Continuous Integration, Test Automation, Acceptance Testing, Architecture & Design, Clean Code, Unit Testing, Thinking about Testing, Test-Driven Development) *Structure:* Teams, Organising by customer Value, Feature teams, Scrum masters, Communities, Organisational Structure	Mini-demos (PS11)
Tools	Free open-source (FOSS) tools, Free Web 2.0 information tools	
Metrics	Not reported explicitly	
	Spotify Model	
Principles	(i) Impact over Velocity, (ii) Innovation over Predictability, (iii) Value delivery over Plan fulfilment, (iv) Community over Structure, (v) Chaos over Bureaucracy, (vi) Trust over Control, (vii) Cross pollination over standardisation, (viii) Alignment enables autonomy, (ix) People are natural innovators, (x) Healthy culture heals broken process	Fluid team structure & Continuous people development (PS139*,**), B2B and product interaction, B2B relationships, Project visibility, & customer satisfaction (PS163)
Practices	Minimum Viable Product (MVP), Validated Learning, A/B testing, Hackdays, Internal Open Source Model, Hack Week, Fail-Friendly Environment, Experiment-Friendly Environment, Weekly Demo, Daily Sync, Culture-focused Roles, Waste-Repellent Culture, Limited blast radius, Release Trains & Feature Toggle, Small & Frequent Releases, Decouple Releases, Self-service Model, Focus on Motivation *Structure:* Squad, Tribe, Guild, Chapter	Pop-up squads & quarterly business review (QBR) meeting (PS139*,**)
Tools	Improvement Boards, Visual Progress, GIT	
Metrics	Not reported explicitly	
	DAD	
Principles	Delight Customers, Be Awesome, Pragmatism, Context Counts, Choice is Good, Optimise Flow, Enterprise Awareness	
Practices	Scrum-based life-cycle, Lean-based life-cycle, The Continuous Delivery:Agile Lifecycle, The Continuous Delivery:Lean Lifecycle, The Exploratory (Lean Startup) Lifecycle, The Program Lifecycle for a Team of Teams	
Tools	DA Toolkit	
Metrics	Not reported explicitly	Business & agile related metrics (PS14*)
	Scrum-at-Scale	
Principles	*Values:* Openness, Courage, Focus, Respect, and Commitment	
Practices	the Scrum Master Cycle (Continuous Improvement and Impediment Removal, Cross-Team Coordination, and Deployment), the Product Owner Cycle (Strategic Vision, Backlog Prioritisation, Backlog Decomposition and Refinement, and Release Planning), Scaled Daily Scrum *Scaled Structure:* Scrum-of-Scrums, Product Owner team, Chief Product Owner, Executive Action Team, Executive MetaScrum Team, Scrum-of-Scrum-of-Scrums	
Tools	Not reported explicitly	
Metrics	Productivity (e.g. change in amount of Working Product delivered per Sprint), Value Delivery (e.g. business value per unit of team effort), Quality (e.g. defect rate or service downtime), Sustainability (e.g. team happiness)	

*Note: * indicates experience reports, ** indicates articles published in IS venues.* s

Table 8: Challenges of Large-Scale Development Methods

Challenges	SAFe	LeSS	Scrum-at-Scale	DAD	Spotify Model	Custom-Built	Scaled Methods
***(C-IC)* Inter-team Coordination**							
(C-IC-1) Synchronising across dynamic and fast-moving teams			PS132*,**			PS17**, PS166, PS179	PS36*, PS149*
(C-IC-2) Communication overload caused by multiple agile layers and ceremonies	PS150, PS157	PS97				PS17**	PS16**, PS36*, PS49,PS96,PS117, PS115*, PS149*, PS174, PS186
(C-IC-3) External distraction (e.g. activities intra- and inter-team)							PS84
(C-IC-4) Enabling end-to-end development***							PS95, PS107
(C-IC-5) Maintaining transparency across high number of fast-moving, adaptive teams and projects							PS36*, PS95, PS107, PS124
***(C-OS)* Organisational Structure Challenges**							
(C-OS-1) Balancing between building generalist and specialist teams							PS25, PS43, PS49, PS92
(C-OS-2) Fluidity of agile roles and no direct mapping from old job roles to the new ones	PS101, PS142, PS162	PS97, PS170			PS139*,**	PS46*, PS121*	PS36*, PS49, PS92, PS149*
(C-OS-3) Complex organisation setup***	PS101, PS142, PS162				PS139*,**		PS50*, PS54, PS87
(C-OS-4) Flow levelling for limited resources/race condition***						PS33**	PS26**, PS140*, PS191
***(C-AR)* Architectural Challenges**							
(C-AR-1) Difficulty in seeing the big picture of the systems***	PS24	PS22				PS17**	PS36*, PS55*, PS117, PS124, PS140*, PS149*
(C-AR-2) Lack of continuous integration and test automation							PS92, PS115*, PS117
(C-AR-3) Lack software security awareness and measure across teams***						PS132*,**	PS16**, PS191
***(C-RE)* Requirements Engineering Challenges**							
(C-RE-1) Difficulties coordinating rapidly changing requirement planning across teams***							PS25, PS41, PS54, PS55*, PS117
(C-RE-2) Capturing value and prioritisation (different customers and teams have different value and prioritisation)***							PS46, PS93, PS117
(C-RE-3) Formulating small, valuable and measurable stories	PS151					PS103**	PS92, PS124, PS130
***(C-CC)* Customer Collaboration Challenges**							
(C-CC-1) Maintaining a constant pace indefinitely***							PS16**, PS110
***(C-MA)* Method Adoption Related Challenges**							
(C-MA-1) Too many agile roles, events, and artefacts	PS104, PS105, PS150	PS170					
(C-MA-2) Scaling agile practices to non-development units	PS101	PS97		PS14*		PS33**	PS121*
(C-MA-3) Lack of proper engineering agile practices and guidelines for agile implementation	PS35, PS38**, PS122, PS162				PS163,PS164, PS166	PS153	PS6, PS34, PS50*, PS54, PS60, PS92
(C-MA-4) Lack of agile maturity	PS35	PS97					PS8, PS49
(C-MA-5) Implementing agile methods to a specific context				PS14*		PS103**, PS121*	PS25, PS40, PS60, PS110, PS187*
***(C-CM)* Change Management Challenges**							
(C-CM-1) Bridging agile culture and mindset at scale	PS24, PS101	PS97, PS170					PS25, PS26**, PS30, PS54, PS95, PS187*
(C-CM-2) Constant change***							PS92
(C-CM-3) Change resistance	PS24, PS91				PS139*,**		PS49, PS92
***(C-TM)* Team-related Challenges**							
(C-TM-1) Lack of ownership of user stories***						PS17**	PS26**, PS181*
(C-TM-2) Over-commitment for faster delivery							PS43
(C-TM-3) Lack of team autonomy	PS24, PS151						PS25, PS92, PS181*
(C-TM-4) Fear of criticism	PS151						
***(C-PP)* Project Management Challenges**							
(C-PP-1) Conflict between long-term planning and the short-term sprint-based planning of agile							PS54, PS140*
(C-PP-2) Alignment to existing processes and stakeholders		PS22				PS121*	PS25, PS43, PS92, PS136
(C-PP-3) Meaningful metrics for performance or improvement	PS66, PS167			PS14*		PS33**	PS6, PS26**

*Note: * indicates experience reports, ** indicates articles published in IS venues,* *** indicates challenges newly identified in our study.

"... if there's too much things going on, it really doesn't tell how things are going. And if the features are, or sub-features are too big, then there's nothing moving." (Kanban, CO103 & CO104, PS124)

The transition from writing full user or system requirements to user stories (C-RE-3) is also challenging during the adoption of ADM (PS103). Vague requirements will lead to vague user stories. Then it is likely that the stories keep changing before a sprint starts. This can cause a waste of time and effort of development teams (PS103).

5.3.5 Customer Collaboration Challenges

To promote sustainable development, agile principles suggest that all stakeholders should maintain a constant pace indefinitely (C-CC-1). However, this has rendered relationship management between the project and client organisation a major challenge. Business unit managers are typically less knowledgable with agile methods, but still need to align their way of working with them. This will create ambiguity in the nature and scope of the project deliverable (PS110).

"What customers have to realise is that we need a day-to-day proximity with them throughout the development cycle but when coding the contacts will likely slow down. We do understand that interacting with IT people takes away from their daily job duties but in the end we are providing them with solutions that will help them in the end to better do their work. So they need to find a balance." (Scrum, CO117, PS110)

5.3.6 Method Adoption Related Challenges

SAFe was designed originally for scaling agile practices in large organisations. Compared to other large-scale methods, it has more roles, events, artefacts, and practices. However, focusing excessively on adopting and tailoring these elements could distract an organisation from achieving its business goals (PS104, PS105, PS150). Similarly, in CO128, one concern reported during the adoption of LeSS was related to the numerous coordination meetings (PS170). Having too many meetings can affect productivity (C-MA-1):

"The PO attends so many meetings that he doesn't have time to write user stories himself ... And if you haven't written them yourself, it's incredibly difficult to accept them." (LeSS, CO128, PS170)

Supporting functions such as sales and marketing should also to be taken care of when the whole organisation transits to agile (PS33, PS101). However, scaling agile practices to non-development units (C-MA-2) is a common challenge in the adoption of SAFe (PS101), LeSS (PS97), DAD (PS14), custom-built frameworks (PS33), and scaled methods (PS110, PS121). This challenge could be intensified if an agile method does not lend itself well to large-scale adoption and customisation in a complex environment, e.g., the unsuccessful scaling of XP in CO100 (PS121).

Despite the increasing adoption of large-scale agile methods in organisations, there is a lack of well-structured approaches or engineering practices (C-MA-3) for the adoption of SAFe (PS35, PS122, PS162) and the Spotify model (PS163, PS164, PS166). For example, the absence of guidelines made the teams of F-Secure (CO38) struggle with internal releases and meeting stakeholders' expectations (PS35). Similarly, six studies on the scaled methods (PS6, PS34, PS50, PS54, PS60, PS92) reported that there was no gradual approaches, proper engineering practices or guidelines for scaling agile methods such as Scrum, XP, or Kanban, which were originally developed for small teams working on small projects, and not intended for large-scale development.

Some studies reported that development teams lack maturity (C-MA-4) with agile methods (PS8, PS35, PS49, PS97), which may be partially due to a lack of guidance provided for agile adoption. PS49 and PS97 highlighted the importance of knowledge about agile methods, including roles and ceremonies, for method adoption.

Implementing large-scale agile methods to a specific organisational context (C-MA-5) is also recognised as challenging (PS14, PS103, PS121, PS25, PS40, PS60, PS110). For example, how the DAD practices could be applied in a more controlled environment, e.g., banking, medical industry.

5.3.7 Change Management Challenges

Several studies reported that a lack of the right mindset and culture (C-CM-1) prohibits organisations from benefiting the whole potential of SAFe (PS24, PS101), LeSS (PS97, PS170) or scaled methods (PS25, PS30, PS54, PS95). The transformation starts with people's mindset to be able to reach the organisational culture (PS24). Embracing transparency and continuously improving based on experimentation require courage and a change of mindset (PS101).

As an organisation attempts to improve its way of working continuously using agile methods, constant change (C-CM-2) is inevitable (PS92). Organisations need to cope with constant and concurrent changes in terms of team structures, processes, tools and metrics.

Both internal and external stakeholders need to be informed in the first place of the reasons for a method's adoption and potential impacts on their daily routines, roles, and responsibilities (PS91, PS139). Not being engaged in a change process could lead to resistance (C-CM-3):

"...even agreeing [on] the dates with the Product Managers for the first PI planning event was hard, as finding suitable time slots from everyone's calendar was not easy, which shows that the importance of this event was not yet understood." (SAFe, CO13, PS91)

When a team has been working for a long period of time, a strong team culture could be established. If it is established before a new agile method is adopted, the adoption at the team level is slower (PS49). Team members are reluctant to change as they already have their internal process in place and thus tend to keep doing the same thing.

5.3.8 Team Related Challenges

The study PS17 showed that the downside of Distributed Scrum was that developers lacked ownership of the assigned user stories (C-TM-1), therefore no commitment to their completion. In the case of CO9, the teams were too relaxed on the deliverables and accepted any delays. The managers and Scrum master did not have control over developers to complete tasks faster (PS181).

The study PS43 reported that, in the case of CO43 in which a scaled method was adopted, the product management was still working in the traditional mode by requesting a long-term feature development plan. This caused overcommitments (C-TM-2) by the development teams and decreased flexibility of the development (PS43).

In the application of scaled methods, some teams may feel a lack of autonomy (C-TM-3) when certain decisions, e.g. sprint length, could not be made by themselves. It makes them feel moving backward, towards the old way of working (PS25, PS92). The lack of autonomy is also perceived in the SAFe adoption. PS151 reported that the teams did not have freedom to choose which features to build. They had been decided in pre-planning meetings.

The study PS151 reported a downside of all joint activities implemented in SAFe, e.g., PI Planning, which is the fear of criticism as teams discuss and show the details of a sprint (C-TM-5). This could cause a loss of clarity as no team member speaks up.

5.3.9 Project Management Challenges

The transition to a scaled method may cause conflict between long-term and short-term sprint-based planning (C-PP-1, PS54, PS140). Backlogs often only provide short-term visibility. Thus, organisations may use their experienced program managers to estimate the subsequent programs. However, this results in more time spent on sprint planning activity.

A lack of alignment with existing processes could be caused by a lack of engagement (C-PP-2) from existing stakeholders (PS121, PS25, PS43, PS92, PS136). For example, middle management should be informed of the changes, e.g. what is happening or why and how they should respond. Otherwise, it creates alienation and ultimately causes management to undermine or ignore the change.

There is a general assumption that a main reason for organisations to adopt large-scale agile methods is to achieve business success (PS66). However, finding the best ways to measure improvement that is expected from adopting large-scale methods (C-PP-3) is challenging (PS14). There is a lack of meaningful measures (e.g. product or process related measures) to capture how organisations progress towards their business goals and values due to the adoption and application of large-scale methods (PS14, PS66).

5.4 Success Factors for the Application of Large-Scale Agile Methods

Table 9 list the factors that can help organisations apply different types of large-scale agile methods successfully.

5.4.1 Management and Organisational

Strong leadership support (SF-MO-1) was needed to adopt SAFe (PS24, PS101), DAD (PS14), Spotify (PS139), and scaled methods (PS16, PS49, PS50) in the case organisations. For example, in CO13 and CO14, the leadership support enabled them to move the whole organisation towards SAFe. Moreover, the introduction of change towards a new method in software development projects usually leads to financial and political pressures (PS14) and a major personnel change (PS139). Thus, visibility and strong support from leadership is important to ensure the success of method adoption.

The support and commitment from leadership (SF-MO-1) during the method adoption and rollout is critical to address any challenges that emerge during the adoption process (PS14, PS87). Teams may not see the benefit of adopting a particular method. However, knowing that the management is continuously working to address the impediments have improved teams' satisfaction towards the method adoption (PS91). Management needs to be engaged and involved in the change (PS80).

> "Their [management] ability to hold firm reinforced the principles of delivering early and often, reducing waste, and sticking to the deadline no matter what. " (ADM, CO12, PS20)

In addition to leadership support, buy-in from all stakeholders (SF-MO-2) are necessary to develop a custom development framework and roll it out across an organisation (PS20, PS51, PS80, PS121). Organisations need to get support from and convince all internal and external stakeholders as the change will also affect them (PS2, PS68, PS139). All their concerns need to be addressed:

> "Most important, ING's executives assured regulators that finance, compliance, and legal functions would continue to be managed in their traditional way." (Spotify Model, CO116, PS139)

One key success factor for implementing the SAFe portfolio is the flexible budget model (SF-MO-4, PS68). As the development progresses from one stage to the next, some part of the budget available is allocated to be based on the previous stage. In this way, management still has funds to allocate in an agile manner to areas with the biggest needs.

In the case of CO116, to keep the structure around customers, the management decided to have a fluid team structure (SF-MO-5) to adapt to do what is the best for customers quicker (PS139). For example, a tribe in charge of daily banking can also handle customer relationships so that they can capture customer issues or needs. After a period of time, these customer relationship tasks are handed over to another tribe in that specialisation.

5.4.2 Process

As organisations start adopting SAFe for their large-scale software development projects, the studies PS24, PS35, PS91 and PS151 suggest them to invest more in the first PI planning meeting (SF-PR-1). In fact, it is one of the critical events to get a buy-in of the change from all stakeholders. People will get a good picture of what is happening and this positively affects their attitudes towards SAFe in general (PS91). The PI Planning event also creates better transparency of overall programs and teams, which creates more opportunities to give and receive help and empowers teams to say no to more work.

In SAFe, having a dedicated full-time team (SF-PR-2), e.g. release train engineers (RTE), is beneficial to lead the coordination among teams (e.g. SoS meetings) and drive continuous improvement by taking care of the metrics (PS24, PS91). RTE can focus on improving the way of working by creating action plans, assigning people, and coordinate the implementation (PS24). In this way, teams are ensured that improvement is ongoing.

Table 9: Success Factors of Large-Scale Development Methods

Success Factors	SAFe	LeSS	Scrum-at-Scale	DAD	Spotify Model	Custom-Built	Scaled Methods
***(SF-MO)* Management and Organisational**							
(SF-MO-1) Strong leadership support and commitment to the agile adoption and roll-out	PS101			PS14*	PS139*,**	PS20*, PS51*, PS80*, PS121*	PS16*, PS49, PS50*, PS87, PS183**
(SF-MO-2) Buy-in from all stakeholders***					PS139*,**	PS2*	
(SF-MO-3) Flexible budget***	PS68*						PS131*,**
(SF-MO-4) Fluid agile team structure***					PS139*,**		
***(SF-PR)* Process**							
(SF-PR-1) Well-prepared planning event	PS24, PS35, PS91, PS151						PS7, PS111*
(SF-PR-2) Full-time and dedicated team***	PS24, PS91					PS20*, PS67*	
(SF-PR-3) An effective mechanism for continuous process improvement	PS24, PS91						PS87
(SF-PR-4) Physical proximity of agile teams***	PS38**	PS11, PS21, PS22					PS39*, PS79, PS190**
(SF-PR-5) Different arenas for coordination over time***	PS38**, PS151	PS22, PS83, PS147					PS146, PS190**
(SF-PR-6) Principles ahead of metrics						PS20*, PS68*	PS87
(SF-PR-7) Balancing agile autonomy with need for oversight					PS139, PS164		
(SF-PR-8) Maintaining transparency across high number of fast-moving, adaptive teams and projects	PS68*	PS170	PS132*,**			PS17**, PS20*, PS51*, PS67*	PS8, PS100*, PS36*, PS37*, PS69, PS77, PS107, PS149*
(SF-PR-9) Customising agile ceremonies, practices and tools support	PS104, PS105	PS21			PS163		PS140*
(SF-PR-10) Architectural guidelines and dedicated architect group***		PS125					PS16**, PS36*
(SF-PR-11) Balanced use of internal software documentation***							PS69
(SF-PR-12) Increase awareness of inter-team dependencies through joint meetings***							PS8, PS43, PS183**
(SF-PR-13) Well-structured adoption approach				PS14*			
(SF-PR-14) Standardise agile practices across teams***			PS132*,**	PS70			PS30, PS81, PS87
***(SF-PE)* People**							
(SF-PE-1) Training and coaching for all on agile adoption	PS24, PS66, PS68*, PS91	PS170		PS14*, PS70	PS139	PS17**, PS20*, PS51*, PS67*, PS80*, PS103**, PS121*	PS13, PS87, PS92, PS95, PS110, PS111*, PS131**
(SF-PE-2) External coach to support method adoption	PS24, PS91, PS101					PS67*	PS87, PS110, PS131*,**
(SF-PE-3) Community of Practice for continuous improvement		PS10			PS164		PS94, PS110, PS159
(SF-PE-4) Sharing a common vision		PS170					PS92, PS95, PS130, PS131**, PS183**, PS187*
(SF-PE-5) Involving and engaging people	PS24, PS91, PS101	PS170				PS2*, PS100*, PS130	
(SF-PE-6) Disciplined teams							PS108*
(SF-PE-7) Trust among teams***		PS11				PS17**, PS67*	PS6
***(SF-TE)* Technology**							
(SF-TE-1) Well-structured information and knowledge sharing systems							PS26**, PS82, PS92
(SF-TE-2) Joint/common infrastructure***				PS14*			PS17**, PS92, PS187*

*Note: * indicates experience reports, ** indicates articles published in IS venues, *** indicates success factor newly identified in our study.*

An effective mechanism for continuous process improvement (SF-PR-3) brings positive results to method adoption (PS24, PS91, PS87). However, to make use of the mechanism, all stakeholders need to have adequate training, mentoring, and coaching as well as their commitment to the continuous process improvement. This also brings long-lasting and sustainable results of the process improvement (PS87).

A physical proximity of teams (SF-PR-4) contributes to the efficient coordination and knowledge sharing for SAFe (PS38), LeSS (PS11, PS21, PS22), and scaled methods (PS39, PS79). When teams are sitting in an open working area, it allows an insight into the work across teams and reduces delays and a lack of communication. For example, progress boards are visible, and it is easier to arrange discussions and meetings. Moreover, it also supports the development of a shared mental model to be able to interpret contextual cues in a similar manner and make decisions for common goals (PS11). Seating arrangements have been suggested to promote transparency and learning in the development chain (PS107). A transparent development process reduces dependencies and increases the planning and coordination between teams in a project (PS69, PS77).

Organisations may use different arenas to improve coordination among teams (SF-PR-5) that suit their situations (PS38, PS151 PS22, PS83, PS147, PS146, PS190). PS22 identified 14 formal and informal coordination arenas, such as board discussion, demo, instant messaging, SoS and wiki. These arenas may change over time depending on the need at the programme level (PS22). This is also the case of SAFe (PS38, PS151). How organisations implement the coordination arenas such as PI planning, SoS and program board may continuously change, depending on the circumstances.

While adopting new methods, teams should focus on principles rather than mechanics such as practices or tools (SF-PR-6, PS20, PS68, PS87). It helps them understand the reason for the change.

Balancing oversight and autonomy (SF-PR-7) is also reported as one of the success factors of the Spotify model (PS139) and the scaled methods (PS6, PS73, PS110). Top-level oversight provides guidance and set ambitious targets and goals, while autonomy gives teams room to decide what and how the work is done. Having a balanced autonomy (e.g. setting own sprint plan, choices of working practices) and oversight (e.g. aligned backlogs, deliveries synchronisation) affects the sense of commitment within a team.

Maintaining transparency across high number of fast-moving, adaptive teams and projects (SF-PR-8) is considered important in the majority of the methods. For SAFe, transparency at the portfolio level improves awareness across an organisation regarding the ongoing development, forces managers to make joint-decision, instead of operating in silos (PS68). At the team level, transparency incentivises teams to deliver high-quality software (PS170) and promotes collaboration, communication and knowledge sharing (PS8, PS100, PS36, PS37, PS69, PS77, PS107, PS149 PS140).

While SAFe provides more ceremonies, practices, and tools than other large-scale frameworks, organisations need to use only the necessary ones and adopt and tailor them to meet their specific business goals (SF-PR-9, PS104, PS105). For example, in the case of CO50, instead of having a sprint review at the end of each sprint, a sprint review was conducted once every three months as it took more time and effort to prepare and attend, but also to allow enough time to develop new features (PS140).

In a large-scale software development project, architectural guidelines (SF-PR-10) are beneficial for development teams as to give methodological guidance to and for management and as to provide a high abstraction level of software (PS36, PS125). Even though architects do not have decision-making authority, they provide inputs to decisions. In the case of CO40, the architecture-focused approach shows a profound impact on enabling continuous delivery (PS153). PS148 suggested that the architectural guidelines (SF-PR-10) should assure the optimal solution from the overall system point of view. To achieve this goal, in the case of CO53, a new structure (CSA) was established to coordinate and support the architectural work together with System and Solution Architect as defined by SAFe.

Balanced use of documentation (SF-PR-11) is perceived important for knowledge sharing across teams and increase project visibility and coordination effectiveness (PS690).

As each team is responsible for their backlog items, they need to identify, recognise and establish a shared understanding of the existence of interdependencies and ways to resolve them (SF-PR-12, PS8). For example, the results of joint planning on specification and prioritisation at the inter-team level help individual teams to specify their high priority tasks in detail and this allows teams to give feedback regarding emergent inter-team dependencies before committing to the next sprint.

In the case of CO7, the organisation had developed a structured enabling program that would guide the transition or adoption to DAD (SF-PR-13, PS14). The program described the project's life cycle from inception to delivery and provided support for roles and clear measurement activities (PS14). Commercial large-scale agile frameworks such as SAFe, DAD or Spotify Model allow each team to use various procedures, practices, or tool suitable to their context. The study PS14 suggested organisations to standardise the way of working (SF-PR-14). The standards enhance quality and product innovation as they enable people to be moved around easily and transferred between projects as needed (PS30, PS70, PS81, PS87, PS130).

5.4.3 *People*

Training and coaching (SF-PE-1) is essential for the adoption of SAFe (PS24, PS66, PS68, PS91) and DAD (PS14, PS70). They have a high pay-off in terms of team productivity (PS13, PS17, PS20, PS80, PS103). Training is useful to communicate with all stakeholders about the why (the reasons for change), and its impact (e.g., generated by new practices, roles and responsibilities, and tools). Participation of higher management in coaching and training sessions is considered to be an effective strategy to demonstrate their commitment to the change process.

Early access to external coaches (SF-PE-2) for guidance, support, and improvement can make an agile transition smoother (PS80). Experienced external coaches are quicker to recognise ways to introduce and embrace changes, and reliable to ensure that agile practices are correctly used (PS110). Thus, at the beginning of the change, they may help organisations to train and coach key personnel and to help with the first PI planning event (PS24, PS91, PS101). These key personnel later become the change agents who can push the change forward and contribute to continuous improvement (PS91). Improvement items need to be implemented and monitored seriously by created action plans and be assigned to responsible persons (PS24).

CoP (SF-PE-3) has proven to help an organisation to coordinate between teams, lead the organisational continuous improvement, share good development practices across the organisation, and handle decisions affecting several teams (PS94, PS159). The knowledge sharing may also happen informally and on-demand across different groups (PS164). It is worth emphasising that these community-based activities and decision making are rooted in experimental culture with fast feedback. Making errors and failures are considered to be learning opportunities.

Common vision (SF-PE-4) is important for establishing a common ground and clarity of goals and direction (PS92, PS170, PS183, PS187). It is a team effort to move all stakeholders towards the common goals. By informing and engaging people (SF-PE-5), the change process gains legitimacy and supports across the organisation (PS6, PS95). Employees should be involved as early as possible to minimise resistance. The motivation to adopt the method needs to be properly promoted and communicated (PS170).

The success of the adoption of scaled methods depends on the discipline of involved teams (SF-PE-6); thus it is necessary to create an environment that encourages team discipline (PS108).

Trust among teams (SF-PE-7) is reported as one key factor for the survival of LeSS in CO6 (PS11). Early in the program, there was a delay in delivery; thus some in the management in Pension Fund wanted to closely monitor the program (PS11). However, the trust from the director empowered the teams to take responsibility, act, and deliver:

> "... the director of the Pension Fund stated: 'Let the people who know how to work, work!' " (LeSS, CO6, PS11)

5.4.4 Technology

In a large-scale project, an adequate infrastructure to support communication, knowledge sharing, and a community of practice (SF-TE-1) can support the development of knowledge networks and social capital (PS82). The infrastructure (SF-TE-2) is necessary to enable end-to-end development processes. This includes joint development tools, test environments, continuous integration, and automated tests (PS14, PS17, PS92, PS187).

6 Discussion

Our study shows that empirical studies on large-scale agile software development is increasing rapidly and steadily in recent years. In comparison to Dikert et al. [6], where only six empirical papers were identified published by 2013, this study identified 191 primary studies published by 2019, and 127 of these are empirical research papers. It is worth noting that, despite the fact that these primary studies are published in both SE and IS venues, our SLR did not find any significant differences in terms of the findings between the two (see Table 8 and Table 9). There are often perceptions that SE and IS are very different; that SE research focuses on technical issues while IS focus on the behavioural and social implications of technology. However, our analysis and the emerging challenges and success factors show that this is not the case. Both fields addressed common issues such as management of large-scale development projects, human factors, organisational issues, and economic aspects of software development and deployment [56], [57], [58], [59]. In fact, it is interesting to note that some researchers published their work in both SE or IS venues.

6.1 Analysing the Use of Large-Scale Software Development Methods

According to a recent industry survey from VersionOne [47], commercialised large-scale methods, such as SAFe, DAD, Spotify model, and LeSS, are among the most popular scaling methods adopted by the respondent companies worldwide by 2019. However, our findings reported in Section 5.1 depict a somewhat different, more nuanced picture. As shown in Figure 2 in Section 5.1, many companies choose to develop their own customised method. These were typically scaled up from agile method implementations that had been proven to be effective at a team level in that organisation, with or without connecting practices to help to scale to large-scale projects. This was understandable before SAFe and other large-scale frameworks appeared and became mainstream in the market. However, the tendency to develop in-house methods has not reduced, even after commercialised large-scale frameworks became available and increasingly adopted. One possible explanation is that organisations are more comfortable scaling what they are already know, rather than a cold switch to a large-scale unknown frameworks. To a certain extent, the primary studies of large-scale development in Nokia over a ten-year period (PS51, PS72, PS79, PS102 and PS151) illustrate this pattern. However, purposefully designed longitudinal studies on the adoption and application of large-scale agile methods are rarely seen in the existing literature. One reason could be that applying agile methods in large-scale software development is a highly complex phenomenon that takes significant research access and resources over a long and unknown period of time. It is possible that researchers lack access to organisations for such a long period, or possibly the long-term funding or resources to do so.

Table 4 shows that the number of studies that address customisation of large-scale frameworks is much smaller than the studies on other types of methods. The majority of those are experience reports. This might indicate a research strand where again practitioners are leading and research is yet to catch up.

The connecting practices reported in Table 5 provide further support to the viability of custom-building own large-scale methods in practice. These practices help tackle the key challenges that companies are confronted with when approaching large-scale software development, such as inter-team coordination, release planning and architecture, and knowledge sharing [6]. The case companies reported in the primary studies developed various practices to tackle these challenges, as shown in Table 5, which could inform other organisations should they choose to grow their in-house large-scale methods. The study by Kalenda et al. [7]

reported 8 scaling practices based on SAFe and LeSS. These are different with our connecting practices, as they are based on scaled up agile methods (non-commercial and customise methods).

Based on the findings in Section 5.1, it can also be argued that there is no defined pattern for approaching large-scale development. Companies have taken very different paths, as demonstrated by the variety of the types of the methods reported in Table 4. In addition, there is no one method that suits a particular type of company. The analysed methodologies were adopted and used by organisations from different business domains, with different method adoption trajectories and usage experiences.

6.2 Analysing Methods by Levels of Abstraction

As far as the authors are aware, there is no previous study that compares large-scale agile frameworks across the levels of abstraction in terms of principles, practices, metrics, and tools. The study by Alqudah & Razali [43] compared these methods by team size, training and organisation type, and was based on the original textbook version of these methods.

Our findings (Section 5.2) reveal that most of the primary studies focused on the practices, tools and metrics of the large-scale frameworks. Very few explored the principles behind them. As shown in Table 7, not all the frameworks have emphasised the underlying principles in an equally comprehensive and explicit manner. This omission of principles in some methods has without doubt constrained empirical researchers who are aiming to study the values or principles underpinning these methods, and from any subsequent comparison between them. One of the most common reasons for agile method failures, even at team level, is that developers often adhere to agile practices without following the spirit of agile thinking [60]. This tendency of "doing agile" over "being agile" looks like continuing in the large-scale movement, given the combination of a lack of detail on some method's principles, combined with a lack of empirical research on the ones that remain. This gap also supports the claim of Rolland et al. [9] that these methods are often built on loose or incorrect values and assumptions, and in addition shows this also applies to the empirical research that has subsequently studied them.

Table 7 shows that seven primary studies offer new emerging principles (PS139, PS163), practices (PS148, PS11, PS139), and metrics (PS14, PS161). However, it must be noted that only two of them were based on rigorous empirical research (PS11, PS163), with the reminder based on experience reports. This could be seen as an opportunity for the research community to actively contribute or even lead the development of large-scale frameworks and methods, to provide a stronger theoretical underpinning for them by examining and explicating the mindsets and principles that truly enable the effective use of the practices from these large-scale frameworks, and in turn to achieve true organisational agility. The studies PS150 and 174 are good examples of this direction. While there was no empirical research on tools, there was at the practice and metrics levels to some extent - new emerging practices and metrics that can fit and complement current large-scale frameworks have been discovered and defined through empirical studies (e.g., PS139, PS123, PS161).

6.3 Challenges in Using Large-Scale Agile Methods

Adopting or customising existing large-scale agile frameworks allows organisations to approach large-scale software development in a more structured and managed manner. However, implementing these methods can also bring new or intensify existing challenges that they are confronted with when developing at a large scale. In our study, we have identified 31 challenges, grouped them into 9 categories and linked them to the different methods for large-scale agile development, as shown in Table 8. They show that the application of large-scale agile methods is challenging and the obstacles presented are both technical and organisational.

Unlike Dikert et al. [6] and Kalenda et al. [7] that study large scale agile transformation in general, we have analysed the identified challenges and reported them by method level. This might be one reason that, even though they overlap with what has been reported in Dikert et al. [6] and Kalenda et al. [7], there are challenges that are reported in our studies only, and some of the challenges identified in their studies were not covered by our findings. A direct comparison of the numbers of identified challenges would be misleading. However, it is worth highlighting that some challenges identified in the previous SLRs which reviewed the literature dated back to 2013 or earlier are captured again by our study of more recently published primary studies, e.g. *(C-OS-2) Fluidity of agile roles and no direct mapping from old job roles to the new ones*, and *(C-MA-2) Scaling agile practices to non-development units*. These are persistent challenges reported repeatedly in the literature.

Most challenges identified in Dikert et al. [6] and Kalenda et al. [7] are largely confirmed by our study. In addition, our study identified 10 new previously unidentified challenges, as indicated with "***" in Table 8. Among them, only one challenge, *(C-CM-2) Constant change*, is reported only by the primary studies published in recent years (after 2016). The other challenges are reported by the primary studies ranging from 2007 to 2019, which means that these are not newly emerged challenges. For example, *(C-AR-2) Difficulty in seeing the big picture of the systems* and *(C-TM-1) Lack of ownership of completion of user stories* are reported in PS17 published in 2007. Another example is *(C-AR-4) Software security* reported in PS132, also published in 2007. Perhaps unsurprisingly, these three challenges are all linked to custom-built methods rather than commercial ones, and the fact this study is the first to include custom-built methods may explain why these were not discovered before. We also mapped each challenges from the specific perspective of the large-scale methods applied in the case companies, and made more explicit linkage between them, as shown by Table 8.

The findings of our study also show that there are not many challenges that are unique to a specific commercialised large-scale method, except *(C-TM-5) Fear of getting critique and be humiliated in sprint planning* which is linked to SAFe (PS151). In comparison, slightly more challenges seem to be only related to custom-built methods or methods scaled from original team-oriented agile approaches, e.g. *(C-*

IC-5) Maintaining transparency across high number of fast- moving, adaptive teams and projects, and *(C-OS-4) Flow levelling for limited resources/race condition*.

6.4 Success Factors in Using Large-Scale Agile Methods

Similar to the identified challenges, we aggregated and reported the success factors at the method level. While some of the factors have been identified in previous studies, this is the first to assign a factor to the individual method where it was uncovered. A success factor was only included if it was clearly shown to be such a factor in the context of a specific method, rather than any abstract argument applying to development generally. In total, we have identified 27 success factors that could be associated to specific methods. Most success factors identified in the previous SLRs are unveiled by our study as well such as *(SF-MO-1) Strong leadership support* and *(SF-PR-8) Maintaining transparency across high number of fast-moving, adaptive teams and projects* [6]. Two success factors, *Keep it simple* and *Recognise the importance of the PO role*, are reported in Dikert et al. [6] but not covered by our study.

A closer inspection of the 12 newly identified success factors from the reviewed primary studies (as indicated with "***" in Table 9) revealed that most of them have been reported in the primary studies that were published between 2003 and 2019, as shown in Figure 2, but not reported in the previous SLRs. In contrast, two success factors, *(SF-MO-5) Fluid agile team structure* related to the Spotify Model, and *(SF-PR-5) Different arenas for coordination over time* linked to SAFe, LeSS and scaled methods, are reported in recent primary studies after 2016.

In comparison to the patterns observed regarding challenges, there are slightly more success factors that are linked exclusively to a specific commercialised large-scale method than to their custom-built counterparts and scaled methods. The success factors linked specifically to the Spotify Model are *(SF-MO-5) Fluid agile team structure*, *(SF-PR-7) Balancing agile autonomy with need for oversight* and *(SF-TE-1) Well-structured information and knowledge sharing systems*. *(SF-PR-3) Continuous improvement* is the success factor linked to SAFe only. *(SF-PR-13) Well-structured adoption approach* and *(SF-TE-2) Joint/common infrastructure* are linked exclusively to DAD. The three unique success factors linked to custom-built and scaled methods are from the *Process* category - *(SF-PR-6) Principles ahead of metrics*, *(SF-PR-11) Balanced use of internal software documentation*, and *(SF-PR-12) Dependencies awareness*.

As compared to previous reviews, our findings linked these factors with large-scale agile methods in a more explicit and specific manner, which can better contextualise these factors from the development method perspective. As discussed in Section 2, we found two reviews in commercial large-scale frameworks [7], [43]. The study by Kalenda et al. [7] reported the challenges and success factors of SAFe and LeSS in an aggregated manner. Thus it is impossible to distinguish which challenges and success factors are associated with each method. It may be the case that the granularity level of the challenges and success factors is different in this study. For example, resistance to change (SC1) seems to have a linkage with all challenges in the Change Management Challenges category in our study. However, only bridging agile culture and mindset at scale are reported to have association with SAFe and LeSS, while change resistance is associated with SAFe only.

As shown in Tables 8 and 9, the majority of the challenges and success factors associated with commercial large-scale frameworks such as SAFe, LeSS and Spotify Model are identified and reported by empirical research rather than experience reports. This indicates that while these methods are originally driven by practitioners, they have been validated by the research community. We have also seen that the majority of the challenges and success factors of scaled methods are evaluated in empirical research than experience reports. This is not the case for Scrum-at-Scale and DAD. We identified three studies of these methods but two of them are experience reports. While the timeliness and importance of these methods to practitioners is evident, the research is still lagging behind.

When one considers the source of the challenges and success factors identified in this study, it is logical that some arise due to the large-scale nature of the development context. For example, inter-team co-ordination (C-IC) has been an issue associated with large-scale projects long before agile (e.g. [61], [62]). Similarly, some challenges and success factors are associated with agile development, even in small (non-large-scale) development environs (e.g. requirements engineering challenge (C-RE) has been identified as a challenge even in the context of 2-5 developers [63]). However, while a challenge or success factor may be primarily attributed to large-scale or agile, our analysis of both sets of literature suggests that all make an appearance in either set, even though it may appear less so in one that the other. Further, even where a challenge may appear in large-scale generally pre-agile (e.g. C-IC), the agile nature of work certainly exacerbates it. For example, the traditional pre-agile response to co-ordination of large teams would be to control from the top down by appointing a set of controllers responsible for ensuring procedures are adhered to using extensive standardised reporting over long periods of time [64], [65]. However, agile principles forbid top down control, extensive reporting and any intolerance of change. Therefore, while co-ordination is a long-standing challenge it is a much more complex one in an agile era, and one that requires different solutions to those of the past.

6.5 Implications for Future Research

This study has a number of implications for researchers. First, it provides the most up to date review of large-scale agile methods for researchers, including 191 papers across 134 case organisations.

Second, researchers are now provided with a set of comparisons between commercial large-scale agile methods (e.g. SAFe, LeSS, Scrum-at-Scale, DAD and Spotify Model), and one that also includes custom-built methods. Previous studies either focused on one method or studied all methods in a collective, aggregated manner. In addition, the analysis compares methods across standard headings of principles, practices, tools and metrics, identifying gaps within each method. Researchers can now aim to develop principles, practices, tools or metrics that effectively fill each gap. Further, research can also examine how gaps in one method could be filled by other methods that are strong in that area. It also provides a standard comparison for researchers to evaluate the effectiveness of each method's set of principles, practices, tools or metrics, and then compare them with the corresponding parts of other methods at that same level of abstraction.

A criticism of method research in general, is that every researcher studies the original textbook method and not take into account the additional improvements added by other researchers that have scrutinised the method since its publication. Table 7 provides researchers with the original method plus a set of extended practices that have subsequently been proposed in the literature. Now, rather than each researcher studying the original textbook method, they can now build on the cumulative, extended work of others, studying not just the original practices, but the additional ones as well.

This study showed, for the first time, most primary studies focused on the application of the practices. Tools and metrics rather than the underlying agile principles, and in fact that the parent methods are often light when it comes to providing detail on these principles. Researchers should seek to address this gap, particularly given the history of agile development where there is a focus on an adherence-based approach, where the team "does agile" rather than "being agile". Table 5 also suggests a potentially new research topic, which involves the study of connecting practices that are required to scale traditional agile methods such as XP and Scrum. While numerous primary studies revealed different connecting practices, dedicated research on connecting practices and an understanding in how they are designed and implemented is yet to be seen.

Our study identifies the challenges and key success factors related to the application of each large-scale agile method. What distinguishes our study from previous work is that we break the challenges and success factors down by method (as shown in Tables 8 and 9). Practitioners or researchers can distinguish between challenges/factors that may clearly been shown to apply to their method, versus those general ones that have been identified in other methods and so may not apply in their own method context. Researchers can now build on these and test the impact of each challenge or success factor and whether they apply to large-scale development generally, or if there are nuances of each or additional ones that emerge in specific contexts e.g. regulated environments, particular sectors, or in high pressured or distributed contexts. This is particularly true where experience reports may have surfaced interesting, but as yet unvalidated findings. We encourage researchers to examine the findings of the experience reports (see the papers marked with an asterisk in Tables 4, 7, 6, 8, 9). Future research could also investigate the severity of the challenges of large-scale agile methods identified in this study. Some challenges, if they are not addressed well, may harm the projects and lead to cancelation or abandonment.

Table 8 and Table 9 show that SAFe, LeSS and scaled methods have received more attention from the research community than Scrum-at-Scale, DAD and Spotify Model. Moreover, primary studies on Scrum-at-Scale and DAD are mainly driven by practitioners than researchers. Therefore, our study calls for more empirical research on these methods to compare and synthesise our findings thus provide complete evidence that can be useful for practitioners.

Finally, we considered both software engineering and information systems literature in the search process. There may be some researchers in either field who are unaware of the relevant work in the other, and so this study may help close the gap between the disciplines.

6.6 Implication for Practice

This study provides many implications for practitioners. First, it provides a single resource for practitioners to learn about the range of large-scale development methods available, rather than sourcing texts on each method separately. Practitioners often ask which method is better, considering this as a binary decision- either a method is adopted or not. Now practitioners can find a link to the core methods, but also links to all empirical research on each method. Also, by comparing across the various method parts, practitioners can see the relative strengths and weakness and the gaps. So they may decide to adopt some or all of the practices of one method and combine them with the practices of another. Or they may supplement practices of a method with tools or metrics of another, where such a combination makes sense. The standardised comparison of methods in this paper allows this to be done in a more considered and informed manner.

Practitioners can consult related primary studies to learn how similar case organisations listed in Appendix C to implement respective methods or frameworks (using Table 4). In such situations, practitioners may take into consideration the challenges and success factors listed in Table 8 and Table 9 to ensure a smooth and successful implementation of the method in their large-scale development projects.

The findings of our SLR show that organisations may approach different paths to choose and adopt a suitable agile method in their large-scale software development projects. However, the journey of scaling an agile method through experimentation, failure and learning may take some time and effort. For example Nokia took more than 10 years to adopt agile methods at scale (XP, Scrum, LeSS). Therefore, practitioners need to be always cautious and take into consideration the complexities of their own development context.

Also, we would hope that by providing the full array of empirical research and experience reports, this would encourage practitioners to reflect, write up and publish their own large-scale agile method journey, either in conjunction with researchers (empirical research) or on their own (via experience reports).

7 Conclusion and Limitations

This paper aims to improve the general understanding of methods used for large-scale agile software development in organisations. This systematic literature review compares the main large-scale agile methods, namely SAFe, Scrum-at-Scale, DAD, the Spotify model and LeSS. In total 191 primary studies across 134 case organisations were identified. It is the first study to analyse and compare each of these methods, as well as custom-built methods, across a set of standard headings, namely the principles, practices, tools, and metrics of each method. For each method, it presents not just the original method specifications but also all extensions and modifications to each method proposed by subsequent empirical research. It reveals a number of theoretical and practical issues in the current literature such as over-emphasis on the practices of commercial large-scale agile frameworks at the expense of their foundational principles. A set of 31 challenges and 27 success factors associated with each method were identified. The study provides researchers with a number of gaps to be addressed across methods. As a result of this study, practitioners can make a more informed decision as to which commercial method or method component or indeed, custom-built method is better suited to their needs based on the findings reported in this study.

In terms of limitations, while agile approaches to large scale development are becoming increasingly prevalent, they are by no means the only approaches for large scale project delivery. In systems engineering for example, there are many projects using other models e.g. [66], [67]. Some of our findings may also be relevant in those contexts. As can be seen from some challenges and success factors, some are specific to, or at least exacerbated in, the context of agile development where the fluidity and dynamism are intentionally inherent.

Also, this study did not consider the levels of method adoption across the papers studied. Given that the adoption of large scale agile methods is often a long and in itself challenging one [10], it is likely that organisations will struggle with different challenges at different points in the adoption process. Moreover, both challenges and success factors will be particularly relevant and exacerbated at various points. Future research could adopt a longitudinal study for example, to examine this over time.

A further limitation of the study is that while many method practices are very clear and operational, others are somewhat vague and open to misinterpretation e.g. the "Be Awesome" practice in DAD. The purpose of this study was not to decipher the meaning of these vague practices, but rather to analyse the empirical papers that studied these practices in an objective way. However, future research could analyse and help strengthen the conceptual depth of these more ambiguous practices by applying an appropriate theoretical lens. In the case of "Be Awesome" for example, a lens from motivation, psychology or innovation theory may be appropriate.

Acknowledgments

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094_P2.

References

[1] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice-Hall, 2002.

[2] K. Schwaber and J. Sutherland, "The Scrum Guide^{TM} – The Definitive Guide to Scrum: The Rules of the Game," 2017.

[3] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley Professional, 2004.

[4] G. B. Ghantous and A. Gill, "Devops: Concepts, Practices, Tools, Benefits and Challenges," in *Proceedings of 21th PACIS*, 2017.

[5] D. G. Reinertsen, *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, 2009.

[6] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and Success Factors for Large-Scale Agile Transformation: A Systematic Literature Review," *JSS*, vol. 119, pp. 87–108, 2016.

[7] M. Kalenda, P. Hyna, and B. Rossi, "Scaling Agile in Large Organization: Practices, Challenges, and Success Factors," *Journal of Software: Evolution and Process*, vol. 30, no. 10, 2018.

[8] J. B. Barlow, J. S. Giboney, M. J. Keith, D. W. Wilson, and R. M. Schuetzler, "Overview and guidance on agile development in large organization," *CAIS*, vol. 29, 2011.

[9] K. H. Rolland, B. Fitzgerald, T. Dingsøyr, and K.-J. Stol, "Problematizing Agile in the Large: Alternative Assumptions for Large-Scale Agile Development," in *Proceedings of 37th ICIS*, 2016.

[10] K. Conboy and N. Carroll, "Implementating large-scale agile frameworks: Challenges and recommendations," *IEEE Software*, vol. 36, no. 2, pp. 44–50, 2019.

[11] M. Laanti, O. Salo, and P. Abrahamsson, "Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation," *IST*, vol. 53, no. 3, pp. 276–290, 2011.

[12] D. Leffingwell, *SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises*. Addison-Wesley, 2018.

[13] C. Larman and B. Vodde, *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*. Pearson Education, 2010.

[14] S. W. Ambler and M. Lines, *Discipline Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. Pearson, 2012.

[15] D. J. Reifer, F. Maurer, and H. Erdogmus, "Scaling Agile Methods," *IEEE Software*, vol. 20, no. 4, pp. 12–14, 2003.

[16] M. Paasivaara, "Adopting SAFe to Scale Agile in a Globally Distributed Organization," in *Proceedings of 12th ICGSE*, 2017.

[17] M. Shameem, C. Kumar, B. Chandra, and A. A. Khan, "Systematic Review of Success Factors for Scaling Agile Methods in Global Software Development Environment: A Client-Vendor Perspective," in *Proceedings of 24th APSEC Workshops*, 2017.

[18] H. Saeeda, H. Khalid, M. Ahmed, A. Sameer, and F. Arif, "Systematic Literature Review of Agile Scalability for Large Scale Projects," *IJACSA*, vol. 6, no. 2, 2015.

[19] J. Klünder, P. Hohl, and K. Schneider, "Becoming Agile while Preserving Software Product Lines," in *Proceedings of ICSSP*, 2018.

[20] A. Putta, M. Paasivaara, and C. Lassenius, "Benefits and Challenges of Adopting the Scaled Agile Framework (SAFe): Preliminary Results from a Multivocal Literature Review," in *Proceedings of 19th International Conference on PROFES*, 2018, pp. 334–351.

[21] M. F. Abrar, M. S. Khan, S. Ali, U. Ali, M. F. Majeed, A. Ali, B. Amin, and N. Rasheed, "Motivators for Large-Scale Agile Adoption from Management Perspective: A Systematic Literature Review," *IEEE Access*, vol. 7, pp. 22 660–22 674, 2019.

[22] B. Kischelewski and J. Richter, "Implementing large-scale agile - an analysis of challenges and success factors," in *Proceedings of 28th ECIS*, 2020.

[23] O. Uludağ, M. Hauder, M. Kleehaus, C. Schimpfle, and F. Matthes, "Supporting large-scale agile development with domain-driven design," in *Proceedings of 19th XP Conference*, 2018, pp. 232–247.

[24] Y. Dittrich, "What Does it Mean to Use a Method? Towards a Practice Theory for Software Engineering," *IST*, vol. 70, pp. 220–231, 2016.

[25] R. Turk, R. France, and B. Rumpe, "Assumptions Underlying Agile Software Development Processes," *Journal of Database Management*, vol. 16, no. 4, pp. 62–87, 2005.

[26] C. Hansson, Y. Dittrich, B. Gustafsson, and S. Zarnak, "How Agile are Industrial Software Development Practices?" *JSS*, vol. 79, no. 9, pp. 1295–1311, 2006.

[27] I. Nurdiani, R. Jabangwe, D. Šmite, and D. Damian, "Risk Identification and Risk Mitigation Instruments for Global Software Development: Systematic Review and Survey Results," in *Proceedings of 6th ICGSE*, 2011, pp. 36–41.

[28] K. T. Yeo, "Critical failure factors in information systems projects," *IJPM*, vol. 20, pp. 241–246, 2002.

[29] M. R. Arizmendi and L. Stapleton, "Failure factors in the control of large-scale business intelligence systems development projects," *IFAC-PapersOnline*, vol. 52, no. 25, pp. 579–584, 2019.

[30] J. F. Rockhard, "Chief Executives Define Their Own Data Needs," *HBR*, vol. 57, no. 2, pp. 81–93, 1979.

[31] S. Matook and R. Vidgen, "Harmonizing Critical Success Factors in Agile ISD Projects," in *Proceedings of 20th AMCIS*, 2014.

[32] J. C. de Almeida Biolchini, P. G. Mian, A. C. C. Natali, T. U. Conte, and G. H. Travassos, "Scientific Research Ontology to Support Systematic Review in Software Engineering," *JAEI*, vol. 21, no. 2, pp. 133–151, 2007.

[33] J. Díaz, J. Pérez, P. P. Alarcón, and J. Garbajosa, "Agile product line engineering – a systematic literature review," *Journal of Software: Practice and Experience*, vol. 41, no. 8, pp. 921–941, 2011.

[34] E. Hossain, M. A. Babar, and H.-Y. Paik, "Using scrum in global software development: A systematic literature review," in *Proceedings of 4th ICGSE*, 2009, pp. 175–184.

[35] S. Jalali and C. Wohlin, "Global software engineering and agile practices: A systematic review," *Journal of Software: Evolution and Process*, vol. 24, no. 6, pp. 643–659, 2012.

[36] F. Kišš and B. Rossi, "Agile to Lean Software Development Transformation: A Systematic Literature Review," in *Proceedings of FedCSIS*, 2018, pp. 969–973.

[37] J. A. Klündeer, P. Hohl, N. Prenner, and K. Schneider, "Transformation Towards Agile Software Product Line Engineering in Large Companies: A Literature Review," *Journal of Software: Evolution and Process*, vol. 31, no. 5, 2019.

[38] P. Lous, M. Kuhrmann, and P. Tell, "Is scrum fit for global software engineering?" in *Proceedings of 12th ICGSE*, 2017, pp. 2–10.

[39] S. Matalonga, M. Solari, and G. Matturro, "Factors affecting distributed agile projects: A systematic review," *IJSEKE*, vol. 23, no. 9, pp. 1289–1301, 2013.

[40] B. Rizvi, E. Bagheri, and D. Gasevic, "A Systematic Review of Distributed Agile Software Engineering," *Journal of Software: Evolution and Process*, vol. 27, no. 10, pp. 723–762, 2015.

[41] R. Sinha, M. Shameem, and C. Kumar, "SWOT: Strength, Weakness, Opportunities, and Threats for Scaling Agile Methods in Global Software Development," in *Proceedings of 13th ISEC*, 2020.

[42] R. Vallon, B. J. S. Estácio, R. Prikladnicki, and T. Grechenig, "Systematic Literature Review on Agile Practices in Global Software Development," *IST*, vol. 96, pp. 161–180, 2018.

[43] M. Alqudah and R. Razali, "A Review of Scaling Agile Methods in Large Software Development," *IJASEIT*, vol. 6, no. 6, 2016.

[44] J. Pernstål, R. Feldt, and T. Gorschek, "The Lean Gap: A Review of Lean Approach to Large-Scale Software Systems Development," *Journal of Systems and Software*, vol. 86, no. 11, pp. 2797–2821, 2013.

[45] W. Alsaqaf, M. Daneva, and R. Wieringa, "Quality Requirements in Large-Scale Distributed Agile Projects - A Systematic literature Review," in *Lecture Notes in Computer Science*, vol. 10153, no. 219–234, 2017.

[46] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," Keele University and Durham University, Tech. Rep., 2007.

[47] *The 13th Annual State of Agile Report*. Collab.NET and VersionOne.Com, 2019.

[48] T. Dybå, T. Dingsøyr, and G. K. Hanssen, "Applying Systematic Reviews to Diverse Study Types: An Experience Report," in *Proceedings of 1st International Symposium of ESEM*, 2007.

[49] M. Kuhrmann, D. M. Fernández, and M. Daneva, "On the Pragmatic Design of Literature Studies in Software Engineering: An Experience-based Guideline," *Empirical Software Engineering*, vol. 22, pp. 2852–2891, 2017.

[50] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software Development in Startup Companies: A Systematic Mapping Study," *Information and Software Technology*, vol. 56, no. 10, pp. 1200–1218, 2014.

[51] T. Dybå and T. Dingsøyr, "Empirical Studies of Agile Software Development: A Systematic Review," *Information and Software Technology*, vol. 50, no. 9–10, pp. 833–859, 2008.

[52] R. F. Paige, J. Cabot, and N. A. Ernst, "Foreword to the special section on negative results in software engineering," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2453–2456, 2017.

[53] T. Saracevic, "Evaluation of Evaluation in Information Retrieval," in *Proceedings of 18th Annual International Conference on RDIR*, 1995, pp. 138–146.

[54] S. Gregor, "The Nature of Theory in Information Systems," *MIS Quarterly*, vol. 30, no. 3, pp. 611–642, 2006.

[55] T. Dingsøyr and N. B. Moe, "Research challenges in large-scale agile software development," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 5, pp. 38–39, 2013.

[56] H. V. Vliet, *Software Engineering: Principles and Practices*. Wiley, 2007.

[57] R. Glass, V. Ramesh, and I. Vessey, "An Analysis of Research in Computing Disciplines," *Communicaitons of ACM*, vol. 47, no. 6, pp. 89–94, 2004.

[58] D. Petkov, D. Edgar-Nevill, R. Madachy, and R. O'Connor, "Informatiion Systems, Software Engineering, and Systems Thinking: Challenges and Opportunities," in *Strategic Information Systems: Concepts, Methodologies, Tools and Applications*. Information Science Reference, 2010, vol. 1, ch. 23, pp. 315–332.

[59] P. Burque and R. E. Fairley, *SWEBOK V3.0: Guide to the Software Engineering Body of Knowledge*. IEEE, 2014.

[60] P. Ranganath, "Elevating teams from 'Doing' agile to 'Being' and 'Living' agile," in *Proceedings - 2011 Agile Conference, Agile 2011*, 2011.

[61] R. E. Kraut and L. A. Streeter, "Coordination in Software Development," *Communicaitons of ACM*, vol. 38, no. 3, pp. 69–81, 1995.

[62] M. Hoegl, K. Weinkauf, and H. G. Gemuenden, "Interteam Coordination, Project Commitment, and Teamwork in Multiteam R&D: A Longitudinal Study," *Organisation Science*, vol. 15, no. 1, pp. 38–55, 2004.

[63] T. Chow and D.-B. Cao, "A Survey Study of Critical Success Factors in Agile Software Projects," *Journal of Systems and Software*, vol. 81, no. 6, pp. 961–971, 2008.

[64] H. D. Benington, "Production of Large Computer Programs," *Annals of the History of Computing*, vol. 5, no. 4, pp. 350–361, 1983.

[65] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," in *Proceedings of ICSE*, 1987, pp. 328–338.

[66] E. Tüzün, B. Tekinerdogan, Y. Macit, and K. Ince, "Adopting Integrated Application Lifecycle Management within a Large-Scale Software Company: An Action Research," *Journal of Systems and Software*, vol. 149, pp. 63–82, 2019.

[67] B. W. Oppenheim, E. M. Murman, and D. A. Secor, "Lean enablers for systems engineering," *Systems Engineering*, vol. 14, no. 1, pp. 29–55, 2011.

Henry Edison is an assistant professor at the Maersk Mc-Kinney Moller Institute, SDU, Denmark. His research examines current and future practices and trends of software development processes, and tailor them to suit different contexts, from startups and new emerging to large and established organisations. He has published in leading journals and conferences in his field including IEEE TSE, JSS, IST etc.

Xiaofeng Wang is an associate professor at the Computer Science Faculty of UNIBZ, Italy. Her main research areas include software startups, agile and lean software development and innovation, and human factors in software engineering. She is actively publishing in SE venues, including IEEE Software, JSS, Empirical Software Engineering, etc. She is also active in serving various SE conferences and workshops. She has collaborated with large companies and software startups on national and European projects.

Kieran Conboy is a professor in Business Information Systems and leads the Lero research group at NUI Galway. He previously worked for Accenture Consulting and the University of New South Wales in Australia. Kieran has published over 100 articles in leading international journals and conferences including ISR, EJIS, and JAIS. His research examines contemporary technology management and design including concepts such as temporality, flow, open innovation and agility. He is an editor of the EJIS and has chaired many international conferences in his field.