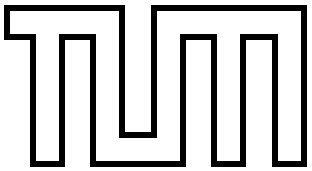


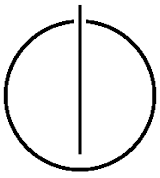
FAKULTÄT FÜR INFORMATIK

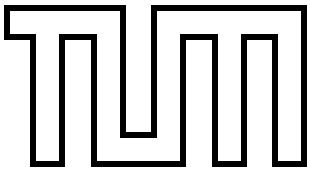
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Survey and Analysis of Process Frameworks for an Agile IT Organization

Binnur Karabacak





FAKULTÄT FÜR INFORMATIK

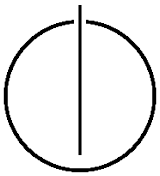
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Information Systems

Survey and Analysis of Process Frameworks for an Agile IT Organization

Überblick und Analyse von Prozessrahmenwerken für eine agile IT-Organisation

Author:	Binnur Karabacak
Supervisor:	Prof. Dr. Matthes
Advisor:	Ömer Uludağ, M. Sc.
Date:	August 16, 2017



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, August 16, 2017 Binnur Karabacak

Abstract

Today's IT organizations are facing a dynamic business environment, which demands that organizations deliver software much faster and better tolerate changing requirements during the project development cycle. To address these challenges, the agile software development, as an iterative and incremental approach, has become an alternative to traditional development methodologies.
Large organizations that are experimenting and succeeding in using process frameworks at team level, like Scrum or Extreme Programming, face their next challenge in scaling these practices across the IT organizations. The process inevitably creates confusion as development teams employ different methods interface with each other. To meet this challenge, several scaling agile frameworks have emerged.
This bachelor's thesis aims to survey and analyze popular process frameworks for supporting IT organizations. This is achieved by conducting a structured literature research and creating a comparison template for juxtaposing the identified process frameworks.

Contents

Outline of the Thesis

Part I: Introduction

CHAPTER 1: INTRODUCTION

This chapter starts by revealing the motivation for scaling agile frameworks. Subsequently, it continues by stating the objectives of the thesis and the underlying research approach for achieving those objectives.

Part II: Foundations

CHAPTER 2: FOUNDATIONS

This chapter presents foundations of large-scale agile software development. Subsequently, large-scale agile software development is described.

Part III: Scaling Agile Frameworks

CHAPTER 3: SCALING AGILE FRAMEWORKS

This chapter presents the identified scaling agile frameworks. Furthermore, it highlights key findings and limitations of related work.

Part IV: Comparison Table

CHAPTER 4: COMPARISON TABLE

This chapter presents the process of creating a comparison template. Based on this template, the Scaled Agile Framework and the Large Scale Scrum are compared. Subsequently, the commonalities and differences are highlighted.

Part V: Conclusion

CHAPTER 5: CONCLUSION AND OUTLOOK

This chapter summarizes the thesis and the results of comparison of the Scaled Agile Framework and the Large Scale Scrum. Additionally, limitations of the conducted research are delineated and a brief outlook for further investigations is provided.

Part I.

Introduction

1. Introduction

In this chapter, we introduce the need for agile software development (ASD) and scaling it to the entire IT organization (see Section 1.1). Following this, the objectives and the corresponding research questions of the thesis are highlighted in Section 1.2. The succeeding Section 1.3 delineates the underlying research approach of the thesis.

1.1. Introduction

Today's IT organizations are facing a dynamic business environment, which demands that organizations deliver software much faster and better tolerate changing requirements during the project development cycle. To address these challenges, the ASD, as an iterative and incremental approach, has become an alternative to traditional development methodologies [118]. An incremental development is a scheduling strategy in which several parts of a system are developed at different times and integrated if completed. Meanwhile, an iterative development is a rework scheduling strategy in which parts of the system are revised and improved [29].

Traditional software development methods are inflexible and unable to respond on customer requests, whereas agile software methodologies provide a set of practices that allow for quick adaptations matching modern product development needs [96].
Compared to traditional project teams, agile project teams provide better return on investment (ROI), enjoy higher success rates, have greater levels of stakeholder satisfaction, deliver higher quality, deliver systems to market sooner [2], require less documentation, and can easily adapt to changing requirements [30]. This does not imply that all agile teams are successful, nor does it mean that all organizations benefit from agile to the same extent [13].

The value of the agile methodologies is proven for small, collocated teams [96]. Organizations that are using agile practices and their qualities, such as less documentation, pair programming and high teamwork at team level, face their next challenge in scaling these practices across the enterprise [37]. As a result, several practices were published in order to scale agile across the team level [84].
There are some publications (see Section 4.1) about scaling agile frameworks, some of which compare the frameworks based on different factors. These comparisons are helpful for organizations as they can form an opinion about which framework suits them best.

1.2. Objectives

This bachelor thesis aims to identify all existing scaling agile frameworks and to give a short description of each. Furthermore, the two most popular ones — the Scaled Agile

Framework® and the Large Scale Scrum — are compared based on a self-created comparison template. This comparison template serves as a basis to compare any other scaling agile framework. Based on these objectives, three research questions are deduced:

1. **Research Question 1 (RQ1):** Which scaling agile frameworks exist?
2. **Research Question 2 (RQ2):** How can scaling agile frameworks be compared?
3. **Research Question 3 (RQ3):** What are the commonalities and differences between the Scaled Agile Framework® and the Large Scale Scrum?

1.3. Approach

The goal of this thesis is to create a comparison template based on which every scaling agile framework can be compared. This will make it easier for organizations to adopt the appropriate framework.

In order to identify material relevant to this goal and assure rigor and relevance of this thesis, a structured literature review approach is applied as recommended by Brocke et al. [134]. The recommended framework consists of the five phases, namely phase I: definition of review scope; phase II: conceptualization of topic; phase III: literature search; phase IV: literature analysis and synthesis; and phase V: research agenda. This framework is used for RQ1, including RQ2 and RQ3. The research approach is illustrated in Figure **??**.
In the first phase, we defined the scope of the review and identified suitable research questions about existing scaling agile frameworks. In the second phase, key concepts were identified, which also provided the opportunity to identify additional relevant search terms (Scaled Agile Organization, Scaled Agile, Agile Organization, Scaled Agile Framework, Agile Framework, and Agile Software Engineering), together with a variety of related concepts, synonyms, and homonyms. The mentioned terms were applied to the subsequent literature search in the third phase. We examined a range of different documentation, theses, conference proceedings, books and Information Systems journals using EBSCOhost, ScienceDirect, Scopus, ACM Digital Library, IEEExplore, SpringerLink, Emerald Insight, and Google Scholar in March 2017. We then used the search terms in electronic full-text search queries. In the first phase, 82 of 759 sources were identified as relevant, given their focus on the topic. In the second phase, additional 39 sources resulted. In the third phase, we searched for documentation on the scaling agile frameworks that we had identified.
In total, 148 relevant sources were obtained. In the fourth phase, we created a comparison table. The presented criteria resulted from the gathered comparison criteria existing in the identified six literature sources. Only the ones that appeared important and meaningful were assumed. Finally, the literature review resulted in a research outcome.

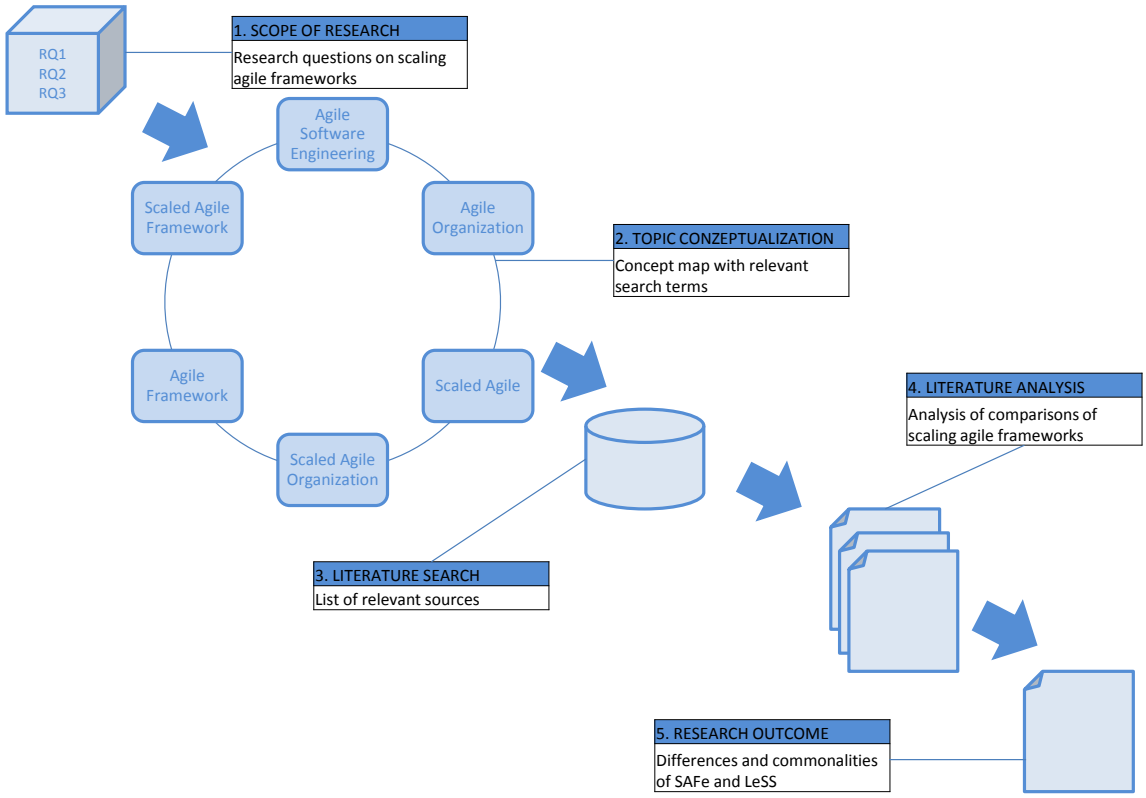


Figure 1.1.: Research approach

Part II.

Foundations

2. Foundations

Section 2.1 provides a basic understanding concerning ASD. In addition, the two most popular agile methods — *Scrum* and *Extreme Programming* (XP) — are described. Section 2.2 provides a basic understanding concerning lean thinking, as it is an essential part of scaling agile development as well. In this context, *Kanban* and *Scrumban* are introduced.

2.1. Agile Software Development

There is no official definition for ASD [13] so numerous definitions exist. But to give an orientation, here is one of several definitions as follows:

"Agile software development is an evolutionary (iterative and incremental) approach which regularly produces high quality software in a cost effective and timely manner via a value driven lifecycle. It is performed in a highly collaborative, disciplined, and self-organizing manner with active stakeholder participation to ensure that the team understands and addresses the changing needs of its stakeholders. Agile software development teams provide repeatable results by adopting just the right amount of ceremony for the situation they face." [13]

Agile development highlights continuous face-to-face communication, short development cycles, learning, and frequent deliveries [53]. ASD is most typically defined by the "Manifesto for Agile Software Development" (Agile Manifesto), which was founded by 17 methodologists in 2001 [1]. Although many of the agile methods were actually introduced earlier then 2001, ASD is often considered to have been born when the Agile Manifesto was written [54]. The Agile Manifesto includes the four values and 12 principles of ASD.
The values are as follows [1]:

- "Individuals and interactions over processes and tools";
- "Working software over comprehensive documentation";
- "Customer collaboration over contract negotiation";
- "Responding to change over following a plan".

That does not mean that processes and tools, documentation, contract negotiation, and planning are not important in agile methodologies, but rather, the focus is on the software. Therefore, individuals and interactions, working software, customer collaboration, and responding quickly to changes seem to have a greater value for agile methodologies [1].
The principles of ASD are listed below [1]:

- Customer satisfaction is achieved through early delivery in a continuous fashion;

- Requirement changes can be adopted even in the final stages;
- Emphasis is always on delivering a product as early as possible, and the delivery can range from a few weeks to few months with an aim to reduce time of delivery;
- Developers and customers need to work together on a daily basis for the entire duration of the project;
- The project teams must have motivated developers working in an environment that harness trust and support;
- Face-to-face conversation is considered the most efficient and effective method within a development team and allows better interactions;
- Working software measures the progress of projects;
- A constant step needs to be maintained indefinitely with developers, sponsors, and users alike;
- Agility must be improved by means of design and technical excellence;
- Simplicity is necessary;
- Self-organizing teams must be available;
- Regular checks guarantee that the progress is going in the right direction and increases efficiency.

These principles facilitate the adoption of agile methodologies for software development [1].

The values and principles of the ASD are represented in agile methods, such as Scrum [110] and XP [49]. Further agile methods include Crystal Family, Dynamic Systems Development Method (DSDM), Feature Driven Development (FDD), Adaptive Software Development [8], Kanban [103], Open Unified Process, and Agile Modeling [13].
Agile methods follow a set of practices and pre-given metrics, and focus on customer collaboration, constant deliveries, lightweight working practices, inflexible development phases [53], iterative development, and small cross-functional development teams [38].
By using agile methods, organizations benefit from increased quality, reduced waste, better predictability, and better morale [109]. In addition, they promise shorter time-to-market, as well as higher flexibility to accommodate changes in requirements and thereby, increase companies' ability to react and respond to evolving customer and market needs [40, 137]. Furthermore, end-to-end responsibility and team autonomy are reported as important characteristics permeating the methods [38].
Scrum and XP are the two most popular agile methods [46, 41, 43, 36, 53]. These meet the need of establishing processes that address the development of systems more quickly and with quality. They use an incremental and iterative life cycle, with short iterations and requirements that can change throughout the development, with extensive participation by the customer [113]. Scrum and XP will be described in more detail in the following section.

2.1.1. Scrum

Scrum was developed by Jeff Sutherland and Ken Schwaber and first presented in 1995. It is an iterative and incremental framework for sustaining and developing complex products [111]. Accordingly, Scrum is most suitable for products and development projects [94].
Scrum is founded on empiricism, or empirical process control theory. Empiricism affirms that knowledge comes from making decisions based on what is known and from experience. Every implementation of the empirical process control maintains three pillars: inspection, adaptation, and transparency. To detect undesirable variances, Scrum users have to inspect Scrum artifacts and progress toward a Sprint Goal regularly. Their inspection should not to be carried out too often. Otherwise it could hinder the current work. When performed by skilled inspectors at the point of work, inspections are most beneficial. If an inspector observes that one or more aspects of a process vary outside the acceptable limits and that the resulting product will be unacceptable, this process must be ceased. To minimize further aberration, a cessation must be made as soon as possible. Scrum specifies four formal events for inspection and adaptation: Sprint Planning, Sprint Review, Sprint Retrospective, and Daily Scrum. These are defined in the next section. Important aspects of the process must be visible to those who are responsible for the outcome. Transparency requires that those aspects are defined by a common standard so observers share a common understanding of what is being seen. When the values of focus, courage, openness, commitment and respect are embodied and realized by the Scrum team, the Scrum pillars of inspection, adaptation, and transparency come to life. The Scrum team members explore and learn those values as they work with the Scrum artifacts, events, and roles [111].

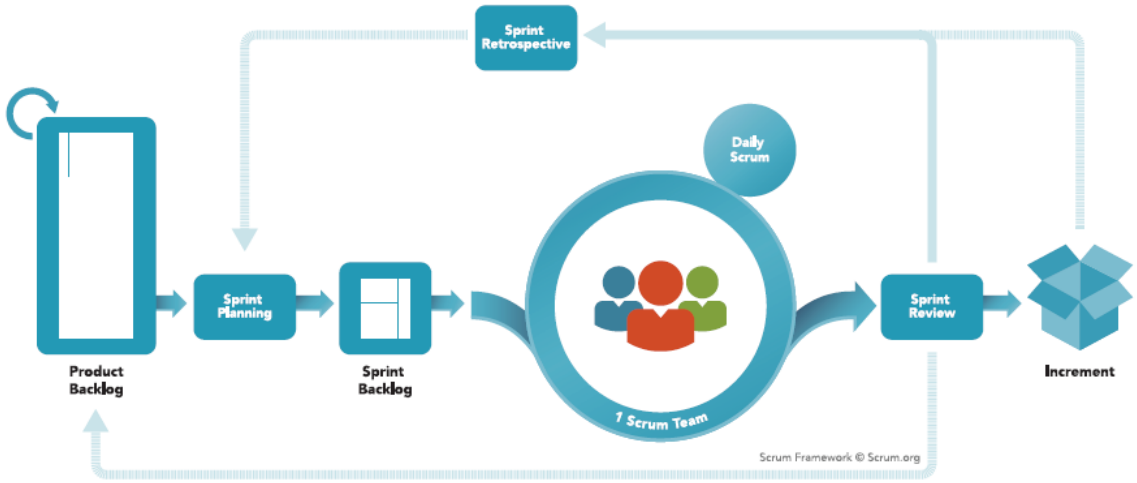


Figure 2.1.: One-team Scrum [112]

Scrum consists of Scrum teams, artifacts, events and rules, thereby each of them has to fulfill a functionality to maintain the success of Scrum.

The Scrum team includes the development team, a Product Owner (PO) and a Scrum Master.
Scrum teams are cross-functional and self-organizing, which means that the team has all the knowledge to achieve the requirements without any help of another team, and they work on their own way rather than being guided by others. To maximize the opportunities for feedback, Scrum teams deliver products incrementally and iteratively.
The PO is responsible for managing the Product Backlog and maximizing the value of the product [111]. A Scrum team has only one PO [3].
The development team consists of three to nine experts, who are responsible for delivering a potentially releasable product Increment at the end of each Sprint. They are working in a cross-functional and self-organizing way.
The Scrum Master hast to make sure that every team member understands Scrum and that the events take place. He also supports the development team by leading them to self-organization and continuous improvement to achieve their goals and by eliminating existing impediments [111].

The Scrum events include the Sprint, Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective. The Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective and the development work are part of a Sprint.
One Sprint can take up to one month, in which the development team has to create a potentially releasable product Increment. For every Sprint, a definition of what is to be built, a design and plan that will guide building it, the work, and the resulting product is stated. The work to be finished in the Sprint is planned at the Sprint Planning, which takes place for eight hours each Sprint.
Every day, the development team holds a 15 minute time-boxed Daily Scrum, where they create a plan for the next 24 hours. There, every team member has to answer the following three questions[111]:

- "What did I do yesterday that helped the Development team meet the Sprint Goal?"
- "What will I do today to help the Development team meet the Sprint Goal?"
- "Do I see any impediment that prevents me or the Development team from meeting the Sprint Goal?"

Daily Scrums improve the development team's level of knowledge and their communications, eliminate other meetings, highlight and promote quick decision-making and identify impediments to development for removal.
At the end of each Sprint, a Sprint Review is held. There, the Increment is checked, and if needed, the Product Backlog will be adapted. The result of the Sprint Review is a reworked Product Backlog that defines the prospective Product Backlog items for the next Sprint.
After the Sprint Review, the Sprint Retrospective takes place. This event gives the Scrum team the opportunity to create a plan, in which the improvements to be implemented in the next Sprint are stated [111].

The Scrum artifacts include the Product Backlog, Sprint Backlog and Increment.
All functions and requirements of a product are stated in the Product Backlog. The Product Backlog is changeable at any time.

The Sprint Backlog consists of requirements stated in the Product Backlog, which has to be fulfilled at the end of a Sprint to create an Increment.
The Increment is the resulting product at the end of a Sprint, which is usable and corresponds to the Scrum team's definition of "done". It contains all the Product Backlog items that are completed in a Sprint.
It is important that everyone has the same understanding of what "done" is. If "done" is a convention of the development organization, everyone has to stick to this definition. If not, the development team has to make an appropriate definition of "done" for the product [111].

2.1.2. Extreme Programming

XP was invented by Kent Beck in 1999. According to Beck, XP is 'lightweight', which means that you only do the essential things to create value for the customer [49]. XP resulted trough the need to increase performance [54]. XP focuses on the general software development process and the tasks, which have to be done during the process, rather than talking about requirement techniques. Furthermore, it accentuates writing tests before coding [93].

The XP team consists of testers, interaction designers, architects, project and product managers, executives, technical writers, users and programmers. In XP, there is no team size specified, and no roles are determined [49].

During the release planning meeting, a release plan is created, which lays out the overall project. Based on the release plan, iteration plans for each individual iteration are created. User stories are exploit to create time estimates for the release planning meeting. They are also preferred to use, rather than a large requirements document. User stories are written by the customers and specify what the system needs to do for them. During the release planning meeting, the development team estimates each user story in terms of ideal programming weeks. During an iteration, the selected user stories will be translated into acceptance tests. To ensure the correctness of a user story's implementation, the customer specifies scenarios to test. A story can have as many acceptance tests as it takes to ensure the functionality works [135].

XP consists of a set of rules [135], values [135, 49], principles and practices [49].
The rules include the management, planning, coding, design, and testing of software [135].
The core values of XP are communication, feedback, simplicity, courage, and respect [49, 135], which guide the development.
There are 14 principles of XP, namely humanity, economics, mutual benefit, self-similarity, improvement, diversity, reflection, flow, opportunity, redundancy, failure, quality, baby steps and accepted responsibility [49].
Beck divides the practices, which XP teams do every day, into two: primary practices and corollary practices. Primary practices include sitting together, whole team, informative workspaces, energized work, pair programming, stories, weekly cycle, quarterly cycle, slack, ten-minute builds, continuous integration, test-first programming and incremental design. On the other hand, corollary practices include real customer involvement, incremental deployment, team continuity, shrinking teams, root-cause analysis, shared code, single code base, daily deployment, negotiated scope contract and pay-per-use. Primary practices can be easily used whereas corollary practices are difficult to apply without first

mastering the primary practices. But not every listed practice needs to be applied to successfully develop software. Which practice to choose depends on the situation. The practices existing in XP make programming more effective [49].

2.2. Lean Thinking

Lean is an evolution of the Toyota Production System, where the emphasis was on continuous improvement, and respect for people. During its evolution, it has become more associated with process improvements aimed at improved efficiency and reduced cycle time [32].

Lean thinking is based on a set of proven mathematical and economic principles that describe the flow of product information within the enterprise. Furthermore, these principles are applied equally well to the supplier and customer elements of the larger business value chain. Therefore, it is broader than the specific agile software methods [79]. Seven principles are subject to lean [85]:

- Eliminate waste
- Amplify learning
- Decide as late as possible
- Deliver as fast as possible
- Empower the team
- Build integrity in
- See the whole

Inspired by earlier houses of lean from Toyota and others, Bass Vodde and Craig Larman reintroduced a "house of lean thinking" graphic. The roof, the two pillars, and the foundation provide the philosophical framework for lean software thinking. The product development flow describes the specific lean principles [79].

According to Vilkki [128], lean thinking gives better tools to understand and address the problems of scaling ASD. Lean and agile approaches complement each other in many domains but there are also challenges in combining these two approaches. Lean thinking aligns better with ASD than with reductionist management methods, because lean thinking also looks at systems holistically, and not by reductionism [54].

The goal of lean is to deliver the maximum amount of value to the customer in the shortest possible time frame, while still achieving highest quality [59, 79].
The foundation of lean thinking is management support by leading the organization, by being competent in the basic practices, and by taking an active role in driving continuous improvement. Thus, management support is a key principle of lean thinking, and is one of the major drivers for lean in the software enterprise. Managers and executives are accountable for continuously advancing practices.
A systemic approach is required of lean thinking in order to manage operations across all the components of the enterprise. The facets of a requirements process must be recognized

and optimized, rather than optimizing the behavior of any other function, such as requirements management, or of a role, such as the PO or Product Manager or even an entire agile team or business unit [79].
Lean thinking can be implemented as lean product development, lean software development or lean manufacturing [54].

2.2.1. Kanban

The Kanban method is an evolutionary and incremental way to handle change for organizations [54] and originates from Corbis, where it was developed from 2006 to 2008 [16]. It is founded on the Toyota Production System and its *Kaizen* approach [16] which defines Toyota's basic approach to doing business and means continuous improvement [59]. In this sense, Kanban enables continuous improvement and catalyzes evolutionary change by using Kanban systems that optically highlight improved understanding of existing conditions [105]. Furthermore, the Kanban method is based on lean principles [79] and supports organizations by providing information about which types of changes will and will not be disruptive for them [105]. The Kanban method was intentionally structured in this way to support evolutionary change, which tends to initially manifest as process optimizations. Kanban can support substantially larger and more dramatic managed changes as its organizational capability matures. It is important to know that the Kanban method is neither a software development framework nor a project management framework. Actually, it does not even describe how to do these things. More significantly, the Kanban method must be layered with an existing process [105]. Kanban is best-suited for production support [94].

The concept of Kanban is related to lean manufacturing. In general, it is a signalling system based on Kanban cards (or signs) used in lean manufacturing to plan what to produce, when to produce it, and how much to produce. In software development, Kanban means a virtual or physical whiteboard and tasks to limit the work-in-progress (WIP) in order to create a limited pull system that reveals system operation problems and encourage collaboration so that the system continuously improves [16]. In addition, Kanban reduces lead time and the number of tasks to be done [106]. In this sense, the developers can only "pull a new card", — begin a new work — if another work is finished; this is the definition of a pull system [16]. The Kanban method applies the principles presented in manufacturing to software development [54].

The four principles, which guide the mindset, are as follows [105]:

- Start with your current work;
- Respect the current responsibilities, roles, process, and titles;
- Pursue incremental and evolutionary change at any time;
- Dare to act as leadership at all levels of the organization.

The six practices, which guide for creating fine-grained practices, are as follows[105]:

- "Visualize";
- "Limit WIP";

- ”Manage flow”;
- ”Make policies explicit”;
- Develop mechanisms for feedback at the organizational, workflow and inter-workflow levels;
- ”Improve collaboratively improvement by using model-driven experiments.”

The principles and values existing in Kanban and lean software development are similar to the ones of agile methods [53, 97].

2.2.2. Scrumban

Corey Ladas introduced Scrumban in his book called ‘Essays on Kanban Systems for Lean Software Development’ in 2009 [55]. According to him, Scrumban is a transition method for moving software development teams from Scrum to a more evolved development framework. Scrumban’s entire added capabilities can be applied to a Scrum context in a variety of ways, but do not have to be. Scrumban emphasizes applying Kanban systems within a Scrum context and layering the Kanban method alongside Scrum as a vehicle for evolutionary change, rather than using just a few elements of Scrum and Kanban to create a software development process. Ultimately, Scrumban is about realizing and aiding the competences already constituted in Scrum, as well as providing new competences and perspectives [105]. Scrumban can be implemented at any level of the organization [105].
The term Scrumban is composed of the terms Scrum and Kanban, as it combines the best features of these two approaches for maintenance projects [94]. It uses the prescriptive nature of Scrum to be Agile and the process improvement of Kanban to allow the team to continually improve its process [94]. Consequently, the roots of Scrumban lay in Scrum and Kanban [105]. Therefore, you first have to understand Scrum and Kanban before understanding Scrumban [105].
Scrumban varies from Scrum in the way that certain principles and practices are highlighted. These are as follows [105]:

- Recognizing the role of management;
- Enabling specialized teams and functions;
- Applying explicit policies around ways of working;
- Applying laws of flow and queuing theory.

Scrumban differs from the Kanban Method in the following ways:

- Scrumban prescribes an underlying software development process framework (Scrum) as its core;
- Scrumban is organized around teams;
- Scrumban recognizes the value of time-boxed iterations when appropriate;
- Scrumban formalizes continuous improvement techniques within specific ceremonies.

Kanban brings three main agendas to Scrumban, which highlight creating enduring results from better service delivery while maintaining resilience in changing external conditions: service orientation, sustainability and survivability [105].
Scrumban includes the four principles and six practices of Kanban to achieve its purposes [105]. In Kanban and Scrumban, the use of roles and meetings are optional [64].

2.3. Large-scale Agile Software Development

Agile methods were originally devised for use in small and single-team projects [24]. However, their shown and potential benefits have made them attractive for larger projects and larger companies also in spite of the fact that they are more difficult to implement in larger projects [39] and pose several challenges. The reason, for this, is that agile is software-focused, construction-focused at the expense of delivery, prescriptive at the expense of flexibility, oriented towards small teams in straightforward situations, too narrowly defined, and additionally, the team is focused at the expense of the overall enterprise when using ASD [95].
Larger projects require additional coordination. Especially handle inter-team coordination poses a problem when applying agile to larger projects. Large-scale agile involves further concerns in interfacing with other organizational units, such as product management, human resources, marketing and sales. Furthermore, large scale may cause users and other stakeholders to become remote from the development teams [34]. One considerable difference between small and large scale adoptions is that large organizations have more dependencies between teams and projects. As a consequence, formal documentation is required and thus, reduces agility [82].

By doing a literature research, Dingsøyr and Moer [35] gathered previous interpretations of what large-scale agile is. In the context of large-scale, "size" plays a decisive role; it had been considered in terms of size in persons or teams, code base size, project duration and project budget. The examples of cases that were called "large-scale" included 40 people and 7 teams [90], a code base size of over 5 million lines of code [98], a project time of 2 years with a project scope of 60-80 features [23], and project cost of over 10 million GBP with a team size of over 50 people [21]. Based on their findings, Dingsøyr and Moer [35] conclude to measure large-scale by the number of coordinating and collaborating teams. As a result, they categorized as large-scale 2-9 collaborating teams and as very large-scale over ten collaborating teams.
Dikert et al. [34] identified a number of further studies discussing large- scale ASD and their interpretations of large-scale, of which all of these referred to the number of people involved. The fact that participants of the XP2014 large-scale agile workshop gave very different definitions for large-scale agile development [35] shows that the meaning of large-scale depends very much on the context and the person defining it [34]. To give you an understanding of what large scale agile means, below are some definitions of scaling agile identified in different publications:

"First, scaling the number of involved teams, this is usually what scaling in the context of agile means. Second, scaling up the necessary system engineering activities in the iterations/sprints prescribed by different agile methodologies."[41]

"There are two fundamental visions about what it means to scale agile. The first, tailoring agile solution delivery strategies to address scaling factors such as geographic distribution, regulatory compliance, and large team size is referred to as tactical scaling. The second, adopting agility across your IT department and your organization as a whole, is referred to as strategic scaling. The good news is that many organizations are tactically applying agile techniques at scale and are succeeding in doing so." [10]

"Scale means that it is even more important to understand the fundamental principles behind the Agile Manifesto and apply them to the specific challenges ." [22]

"Enterprise Agile development, or scaled Agile, is the next horizon of Agile methodology adoption. Enterprise Agile, or scaled agile, refers to the design and implementation of Agile methodology (irrespective of the specific flavor of Agile, e.g. Scrum, Kanban, XP) for use on large scale IT programs or large projects at an enterprise level." [84]

Part III.

Scaling Agile Frameworks

3. Scaling Agile Frameworks

The focus now includes scaling the agile methods for a variety of applications [44]. In an attempt to scale the advantages of agile methodologies, various frameworks have been proposed to provide guidance for scaling agile development across the organization. One of the well-known models is the Scaled Agile Framework®. Some researchers agree that guidelines and adaptions of agile practices might be necessary when scaling along the dimensions of complexity, project size, and dispersion of team members. As a method is scalable only if it can be applied to problems of different sizes without fundamentally changing it, scalability of agile is arguable [123].

3.1. Scaling Agile Frameworks

During the literature analysis and synthesis phase of the literature review, 148 distinct information sources were analyzed. Table 3.1 provides an overview of the distinct literature source types.
Papers (82 in total) represent the majority of the relevant information sources followed by white papers (totaling 19). Furthermore, 13 information sources are articles, 11 are websites, ten are presentations, and six are books. Handbooks, dissertations, and master theses form only a small part of the relevant information sources.

Literature source type	
Characteristic	Count
Paper	82
White paper	19
Articles	13
Websites	11
Presentations	10
Books	6
Master's thesis	4
Handbooks	2
Dissertation	1
Total	148

Table 3.1.: Overview of literature source type distribution

During the literature research, 20 scaling agile frameworks were identified. These facilitate the adoption of agile beyond the team level.

Below, the identified scaling agile frameworks:

- Crystal Family
- Dynamic Systems Development Method
- Scrum of Scrums
- Enterprise Scrum
- Agile Software Solution Framework
- Large Scale Scrum
- Scaled Agile Framework 4.0®
- Disciplined Agile 2.0
- Spotify Model
- Mega Framework
- Event-Driven Governance
- Recipes for Agile Governance in the Enterprise
- Matrix of Services
- Scrum at Scale
- Enterprise Transition Framework
- Scaled Agile Lean Development
- eXponential Simple Continuous Autonomous Learning Ecosystem
- Lean Enterprise Agile Framework
- Nexus
- Fast Agile

Below, I will briefly describe each scaling agile framework and provide some information concerning the methodologist, the publication date, the category, and the organizations that were built upon the frameworks. In addition, I will provide information concerning the adoption, such as the number of contributions that were made regarding each framework, the number of case studies, and the availability of documentation, training courses, and communities.
The Scaled Agile Framework® (SAFe®) and the Large Scale Scrum (LeSS) are especially mature, because they are cited very often in the literature, they describe many real-world use cases, and they also fulfill the other adoption criteria [89]. Because of this, I will provide more details about them in the following sections.

3.1.1. Crystal Family

Crystal is a set of methods presented in 1992 by Alistair Cockburn [89]. It is also called Crystal Family, as it is a family of methodologies. Each crystal has a different hardness and color, corresponding to the criticality and project size. Crystal methods include inter alia clear, yellow, orange, orange web, blue, red, and magenta. The crystal clear, for example, is suited for up to 6 persons, crystal yellow for up to 20, crystal orange for up to 40, crystal orange web for up to 50, and crystal red for up to 80 persons working on a system. Each method is people- and communication-centric, gets adjusted to fit its particular setting, and works from the project tolerance level and the bottleneck activities to an answer that matches the project ecosystem. Furthermore, the project developed in each method must use incremental development, with increments of four months or less, and the team must hold pre- and post-increment reflection workshops [28]. The methods included in the Crystal Family focus on frequent delivery, close communication, and reflective improvement. These properties should be included in all projects. To get further into the safety zone, the teams also use other properties, such as personal safety, focus, easy access to user experts, technical environment with automated tests, configuration management, and frequent integration. Delivering running and tested code to real users is a must in every project. Frequent delivery includes advantages for teams, inter alia getting critical feedback on the rate of progress of the team, and getting to debug the development and deployment processes. To support the close communication, the teams sit in the same room. If one person asks a question, the others can contribute to the discussion or continue with their work. In a meeting, the team inspects their work, lists what is and what is not working, reflect on what might work better, and apply those improvements in the next iteration [27].

Nineteen contributions were made about Crystal and one enterprise is currently using it. You can find documentation and a blog, but there are no training courses available [89].

3.1.2. Dynamic Systems Development Method

The Dynamic Systems Development Method (DSDM) framework was published in 1994 by Arie van Bennekum [89]. It is a proven framework for agile project management and delivery, helping to deliver results effectively and quickly. Over the years, it has been applied to a wide range of projects from small software development up to full-scale business process change.

The eight principles of DSDM bring the agile values to life by guiding the team in the attitude it must take and the mindset it must adopt in order to deliver consistently whilst still remaining flexible. These principles include focusing on the business need, developing iteratively, delivering on time, never compromising quality, building incrementally from firm foundations, collaboration, communicating continuously and clearly, and demonstrating control. The DSDM uses both an iterative and incremental approach.

The DSDM process model consists of a framework which shows the DSDM phases and how they relate to each other. It is used by each project to derive its lifecycle and has four main phases: feasibility, foundations, evolutionary development and deployment. These are preceded by the pre-project phase and followed by the post-project phase. In total, this results six phases.

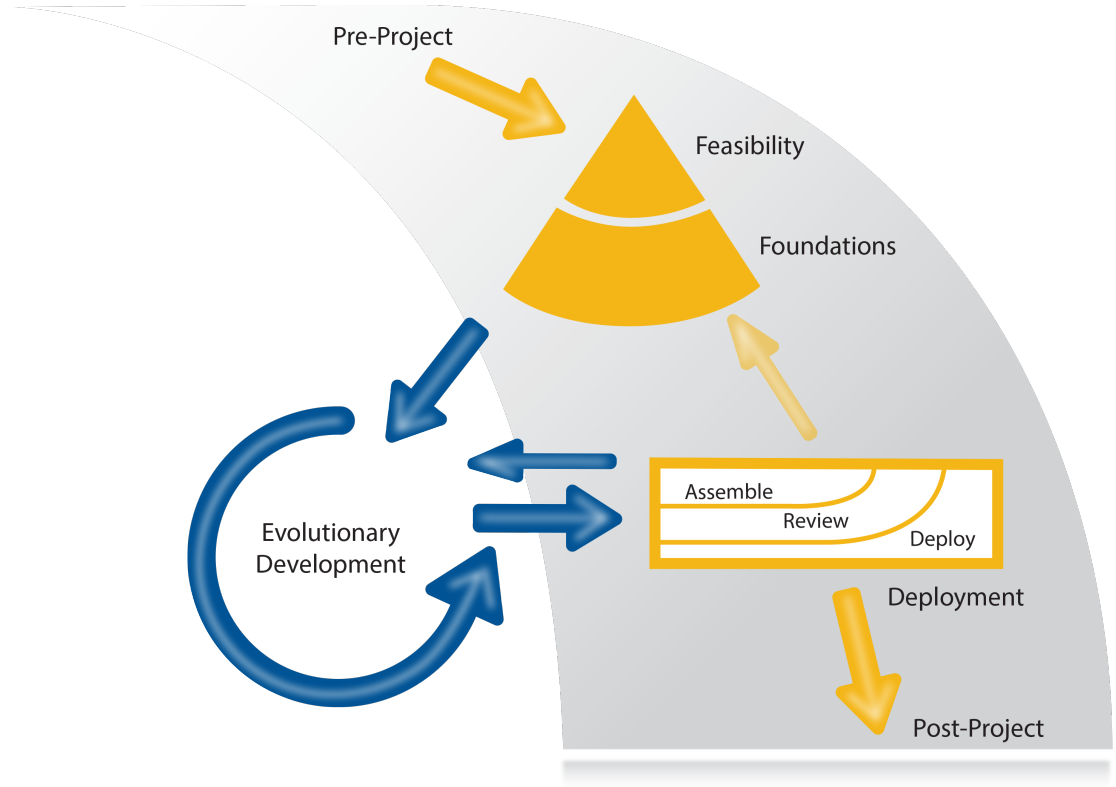


Figure 3.1.: The Dynamic Systems Development Method process [125]

The pre-project phase ensures that only the right projects are started, and that they are set up correctly.
The feasibility phase intends, primarily, to determine whether the proposed project is likely to be feasible from a technical perspective and whether it appears cost-effective from a business perspective.
Foundations aim to understand the scope of work, how it will be carried out, by whom, when and where. In addition, the foundations phase determines the project lifecycle by establishing how the DSDM process will be applied to the specific needs of this project.
The evolutionary development phase aims to evolve the solution. It requires the solution development team(s) to apply practices such as timeboxing, iterative development, and MoSCoW prioritization, together with modelling and facilitated workshops, to converge over time on an accurate solution that meets the business need and is also built in the right way from a technical viewpoint. MoSCoW is a technique for understanding and managing priorities, in which the letters indicates *Must Have, Should Have, Could Have*, and *Will not Have this time*.
The deployment phase aims to bring a baseline of the evolving solution into operational use. This includes three main activities: assemble, review and deploy.
After the final deployment, the post-project phase checks the extent to which the expected

business benefits have been met.

In the context of scaling, the project organization can easily be refined to support multiple teams, with key roles acting as directors and coordinators across the teams. In order to support a more complex project structure, products, such as the solution architecture definition, management approach definition, development approach definition, timebox review records and the delivery plan can be made more elaborate and more formal than would be appropriate for smaller projects [31].

The DSDM Consortium was built upon the DSDM framework. Thirty two contributions were made and four enterprises are currently using DSDM. Documentation, as well as communities, and training courses are available [89].

3.1.3. Scrum of Scrums

The Scrum of Scrums (SoS) was presented in 2001 by Jeff Sutherland and Ken Schwaber [89]. SoS, as a piece of the Scrum at Scale approach to scaling Scrum, is a mechanism used to coordinate dependencies and the release of products across multiple teams. Scrum limits the number of communication ways needed to transmit information relevant to the success of the enterprise, by scaling fractally. SoS is the same as the Daily Scrum at the team level. The difference is that SoS is a virtual team composed of representatives from a number of individual Scrum teams that collaborate to integrate and ship a product.

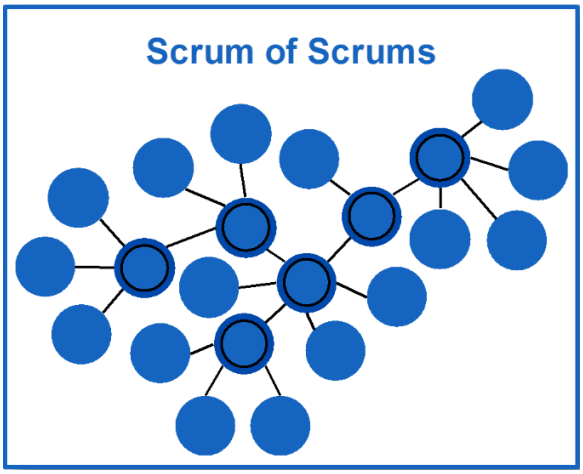


Figure 3.2.: Scrum of Scrums, Source: based on [4]

The Scrum Masters and other required roles meet and communicate impediments, progress, and any cross team coordination that needs to happen by answering for the teams the same three questions used in the Daily Scrum. The minimum viable release team for products and product lines is a properly executed SoS. It aims to remove the waste introduced into many Scrum implementations by integration teams, release teams, and release management teams introduced by scaling frameworks. Outside the SoS meeting, volunteers from the meeting deal with eliminating operational impediments that are identified as occurring during the release and deployment process, which is equivalent to Scrum team members working together in a Sprint. According to the role of management, the SoS Master is re-

sponsible for delivery. Therefore, the SoS Master is usually a more senior person, often at the Director of Engineering or higher level. SoS is not the Enterprise Action, but rather it may refer company issues to the Executive Action Team, although the SoS deals directly with operational issues.
In SoS, there are two additional scaling roles:
The *Executive Action Team* (EAT) is a Scrum team consisting of top level executives. For the entire organization the Product Owner of this team should be the Chief Product Owner (CPO). The team members are ideally members of different organizational departments that help execute the vision of the organization as expected by the CPO. The team needs to be cross-functional, self-organized, and self-managed, just like any Scrum team.
The *Meta Scrum* indicates how the Product Owner (PO) role scales in Scrum and comprises a virtual team made up of four to five POs that coordinate epics, product lines, and product releases. An epic, which is created by the EAT, is most likely too big to be completed by one single Scrum team. To ensure coordination, the POs from all the teams working on one large epic get together when necessary. The get together is usually coordinated by a scaled network of Meta Scrums including the EAT. It is the EAT's work to create the organizational backlog that feeds large epics to the network of POs [4].

Scrum Inc. is built upon the SoS mechanism. Thirty four contributions were made and two enterprises are currently using SoS. There are documentation and communities available, but no training courses [89].

3.1.4. Enterprise Scrum

The Enterprise Scrum (eScrum) framework was presented in 2002 by Mike Beedle [89]. EScrum adapted and extended Scrum based on the abstraction, the generalization, and the parameterization that can be used in a scaled generic way for any management purpose. Concerning the abstraction, eScrum describes better what Scrum is. Furthermore, eScrum uses the same concepts in Scrum, but in a more general way. For instance, Scrum is a 2-level subsumption architecture, whereas eScrum is a n - level subsumption. Finally, eScrum is a parametrization of Scrum because it adds parameters to explicitly track things [19]. EScrum provides 80 further parameters, which were abstracted primarily by observing what people had already done in the field as they used Scrum for different purposes. These parameters extend, customize, and apply the Scrum concepts to different scaling or adding techniques and domains, and make eScrum a true framework, as the customizable parts are now visible and explicit.
Structure parameters allow eScrum scaling into larger organizations by connecting multiple eScrum instances in different ways. These instances include subsumption, collaboration, delegation or centralized [19].

EScrum has the same roles with the same functionality as does Scrum, but they have different names.
The Product Owner is called Business Owner and the Scrum Master is called Coach. The team is called team in both approaches. Besides the roles, other concepts are the same in Scrum, but are renamed in eScrum. In eScrum, the Product Backlog is called Value List, which is a list of things that add value when done. The Product Backlog Item is called Value List Item. Sprints are called time-boxes cycles. Cycles can be recursive without any limitation. For example, you can have a yearly cycle, which contains quarterly cycles that

contain two week cycles. The Release Planning is naturally included for all cycles and at all levels.
Scrum has the Burn Down charts, whereas in eScrum any type of report can be chosen [20].

Upon this framework the Enterprise Scrum Inc. is built. Ten contributions were made, but no information about enterprises using this framework is available. Documentation, as well as communities, and training courses are available [89].

3.1.5. Agile Software Solution Framework

The Agile Software Solution Framework (ASSF) was presented in 2007 by Asif Qumer and Brian Henderson-Sellers [89]. Its elements can be classified in terms of tools and the agile conceptual aspect model. The agile conceptual aspect model represents the aspects of knowledge, governance, and method core, which are linked to business via business value or a business-agile alignment bridge.
The method core and abstraction elements of ASSF represent six aspects of an ASD methodology, namely agility, people, process, product, tools and abstraction. This set of aspects provide a mental-model or vision-guiding for an agile methodology.

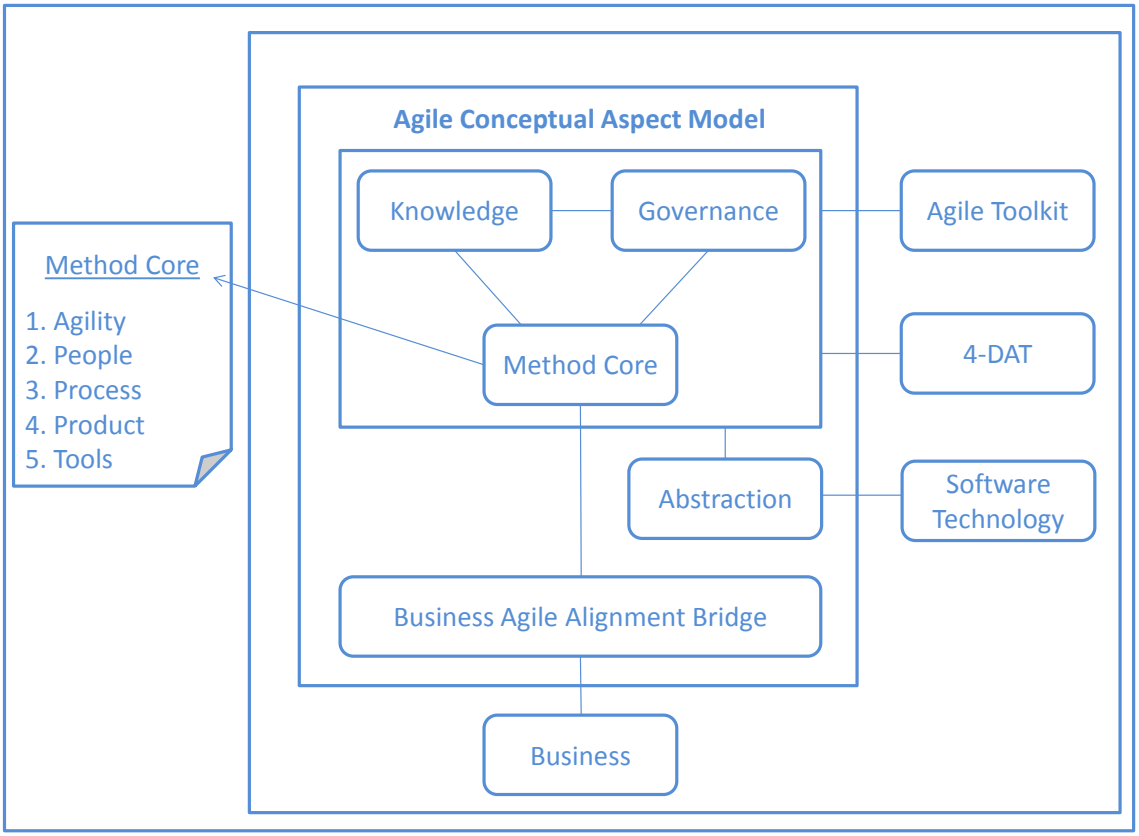


Figure 3.3.: The main components of the Agile Software Solution Framework, Source: based on [101]

The Knowledge Cell is used to manage and engineer the knowledge related to ASD. An agile knowledge engineering and management approach [17] should be integrated with an ASD approach, in order to manage and capture related knowledge and to use it for performance improvement, learning and decision making in an ASD environment.
Governance provides a mechanism for a strategic IT-business alignment. This makes it possible to acquire maximum business value delivered by the consumption of IT resources, but it has not been discussed in the context of ASD organizations. Therefore, a model for accountability, responsibility and business value governance in the context of agile development [101] is developed to introduce sufficient control, discipline, and rationale to scale up ASD methods for large and complex projects.
The business-agile alignment bridge has not been investigated to any great extent by the agile community. It has an impact on both the construction and application of agile methods in terms of the business value delivered to customer, process, team, product, and workspace. This bridge aligns ASD goals and business goals.
The agile toolkit facilitates the construction and valuation of multiabstraction, which is a mix of different abstraction-mechanisms, such as object-oriented, agent-oriented, and serviceoriented.
The four-dimensional analysis toll (4-DAT) will facilitate the examination of agile methods from four dimensions: agility characterization, agile values characterization, method scope characterization, and software process characterization. The only currently quantified dimension of the 4-DAT is the agility characterization, which permits a numerical evaluation of the degree of agility at both the large scale and the small scale. The large scale refers to the process level, meanwhile the small scale refers to the method practices level.
Business refers to a software development organization. The policies, business goals, strategies, and the existing culture of a software development organization have a potentially large impact on the construction, adoption, execution, and governance of any ASD method [101].

Reagrding ASSF, three contributions were made and two enterprises are currently using this framework, but neither documentation, nor communities, nor training courses exist [89].

3.1.6. Large Scale Scrum

Large Scale Scrum (LeSS) is a framework that was presented in 2008 by Craig Larman and Bas Vodde [57]. It is not a framework that applies Scrum at the team level and then adds additional scaling processes. Rather, it is Scrum scaled on all levels. Basically, LeSS is one-team Scrum applied to many teams who are working together on one product. LeSS is about understanding how to apply the principles, elements, purposes, and elegance of Scrum in a large-scale context as simply as possible [62].
LeSS includes the following components [62]:

- Rules define the foundation and key elements of the framework. Similar to Scrum, the focus lies on the structure of teams, the development process (e.g., Sprints), the definition of the requirements of the product (e.g., backlog), and the roles within the team (e.g., Scrum Master).

- Principles provide answers as to how to apply LeSS in specific enterprise contexts.
- Guides support the adaptation of the rules and experiments by providing best practices and tips.
- LeSS fortifies teams to experiment, to fail, and to learn new concepts.

LeSS comprises two different large-scale Scrum frameworks, namely Small LeSS and LeSS Huge. Small LeSS is suited for up to eight teams of eight people on each team, whereas LeSS Huge is suited for more than eight teams, and has means for hundreds of people working on a single product. In this context, the word *LeSS* is equal to large-scale Scrum in general and to the smaller LeSS framework [62].

LeSS has the same artifacts as Scrum, namely one potentially shippable product Increment at the end of each Sprint, one Product Backlog, and a separate Sprint Backlog for each team [62].

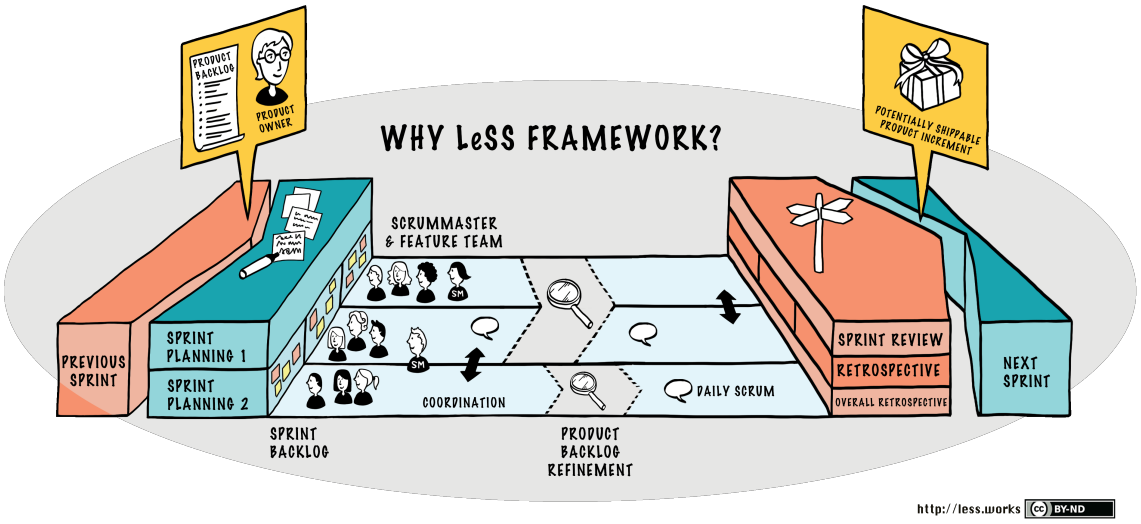


Figure 3.4.: The Large Scale Scrum Framework[130]

As well, the events, such as a common Sprint for the whole product, the Sprint Planning, the Daily Scrum, the Sprint Review, and the Sprint Retrospective, are present in LeSS.
New to LeSS is the Overall Retrospective meeting, whose purpose is to discuss cross-team, organizational, and systemic problems within the organization. The PO, Scrum Masters, team representatives, and managers take part in this meeting. Conceptually, the Overall Retrospective takes place right after the team Retrospectives. But as the team Retrospectives are often at the end of the day, at the end of a Sprint, people are often exhausted or do not have time to continue with another Retrospective meeting. Therefore, the common way is to have the Overall Retrospective at the beginning of the next Sprint.
Another difference between Scrum and LeSS is the Sprint Planning. This planning consists of two parts. Sprint Planning One is a meeting for all of the teams together, where they decide which team will work on which items. The focus is on the selection of ready items from those offered by the PO, defining of the Sprint Goal, and wrapping up lingering

questions. Sprint Planning Two is a separate meeting for each team, where each creates the plan for getting the items to "done" during the Sprint [62].

Concerning the roles in Small LeSS, there is one PO for all teams and a Scrum Master, who is responsible for one to three teams. Conceptually, the PO role in LeSS is the same as in one-team Scrum. However, at scale, the focus is rather on keeping an overview and ensuring the maximum return on investment (ROI) on the product [62].

In LeSS Huge, there is one PO and at least two Area Product Owners (APO). LeSS Huge introduces additional scaling elements, which are required to manage hundreds of developers in large enterprises, and a new concept, namely Requirement Areas (RAs).
Each RA includes four to eight teams [62]. RAs comprise major areas of customer concern from a product point of view and may grow or dwindle over time in order to match product needs. The RAs follow the same Sprint rhythm and aim for continuous integration across the entire product [89].
Furthermore, there is an Area Product Backlog (APB) in LeSS Huge. The APB is a view into the Product Backlog based on the RA. Each Product Backlog item belongs to one Area Backlog, and the other way around. If necessary, Area Backlog items are defined, prioritized, and split by the APO. The APO focuses on one APB and acts in essentially the same way as the PO, but with a more limited, and customer-centric perspective [62].

Through the elimination of project manager roles and a traditional team lead, LeSS also introduces feature teams, which are cross component and cross-functional [61]. Like with cross-functional teams, feature teams focus on the customer and remove the barriers between customers and actual users [62].
In comparison with the traditional Scrum approach, the LeSS framework changes the structure of Sprint Planning meetings. Here, two members of each team plus the one overall PO decide which Product Backlog items to work on, whereas in standard Scrum, the rest of the agile team also participates. In the case of a contention over a Backlog item, the PO mediates between teams.
Furthermore, Sprint Review changes to a single meeting for all agile teams, which is limited to two team members per agile team. Two more changes are determined [124]:

- The purpose of the inter-team coordination meeting is to increase information sharing and coordination. It can be held frequently during the week and can take various forms, including a multi-team Daily Scrum, open space, Scrum of Scrums formats, or a town hall meeting.
- The focus of the joint light Product Backlog refinement meeting is on refining Product Backlog items for upcoming Sprints. Only two representatives per team attend. The meeting should not exceed 5% of the Sprint duration.

Upon the LeSS framework, the LeSS Company B.V. was built. Twenty nine contributions were made, and 22 case studies are reported. Documentation, as well as communities, and training courses are available [89].

3.1.7. Scaled Agile Framework® 4.0

The Scaled Agile Framework® (SAFe) was first presented in 2011 by Dean Leffingwell [89]. Currently, SAFe® released the latest version 4.5 [76]. Nevertheless, the version 4.0

is described hereinafter. Subsequently, the differences between the SAFe® 4.5 and 4.0 are highlighted.

3.1.7.1. Description of SAFe® 4.0

SAFe® enables implementing agile practices at enterprise scale. The framework focuses on highlighting the roles, activities, and artifacts that are necessary to scale agile to teams, programs and enterprises [118].

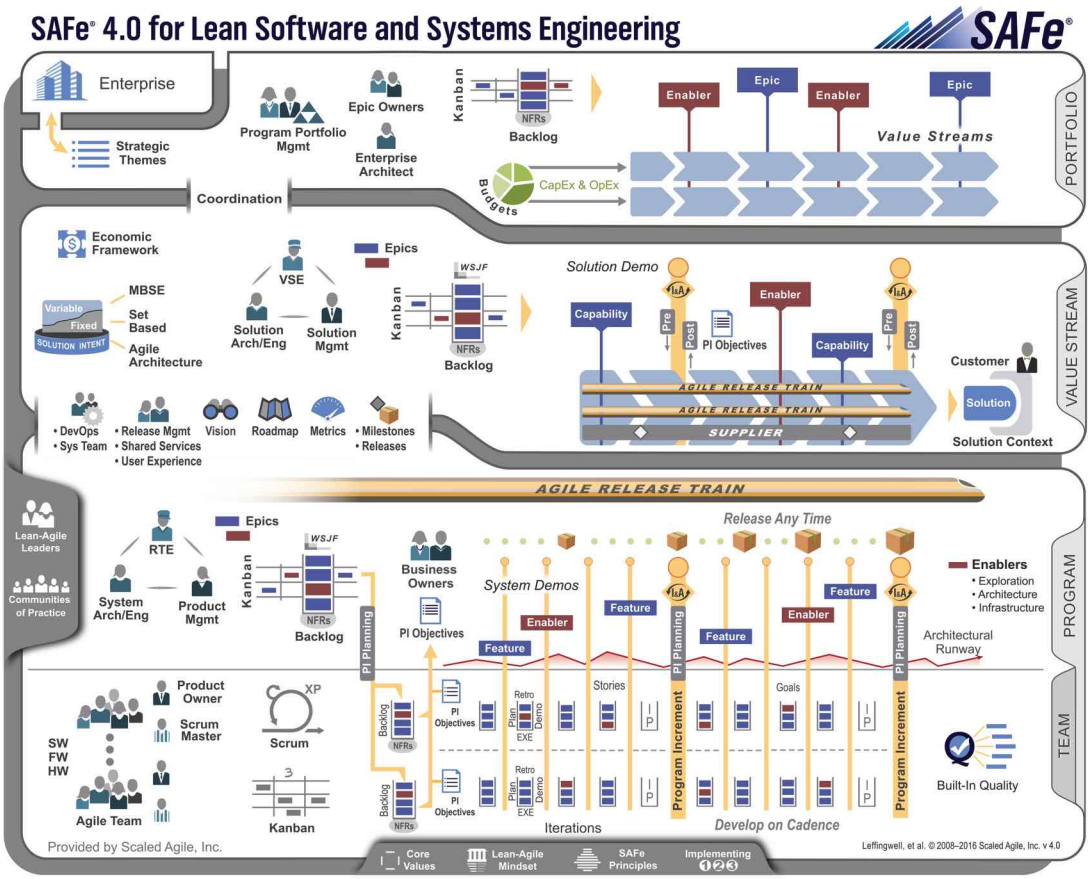


Figure 3.5.: The Scaled Agile Framework®: *The Big Picture* (Reproduced with permission from © 2011-2017 Scaled Agile, Inc. All rights reserved.) [77]

SAFe® follows a combination of existing agile and lean principles, and combines them into a method for large-scale agile projects. Furthermore, it provides guidance and training for scaling agile development across the portfolio, value stream (optional), program, and team levels. Besides these four layers, SAFe® can be additionally configured with a foundation layer. Because the framework is scalable and modular, it allows each organization to adapt it to its own business model [80].

- Team level:

SAFe® fundamentally consists of agile teams. The team level comprises the roles, events, activities, and processes through which agile teams build and deliver value. Each agile team consists of 5-9 agile team members, a Scrum Master, and a Product Owner (PO) and is responsible for defining, building, and testing small pieces of new functionality from their Team Backlog.
To synchronize work with other teams, each use a common iteration cadence, which allows the entire system to iterate simultaneously.
Each agile team employs ScrumXP or team Kanban. ScrumXP is a process, which combines Scrum project management practices and XP technical practices.
The development timeline is divided into a set of iterations within a Program Increment (PI). The PI is a fixed timebox that synchronizes delivery, planning, and reviews within an Agile Release Train (ART) . Each iteration begins with an iteration planning, where the team members decide how much of the Team Backlog they can commit during an iteration. The work to be finished during an iteration is summed up as PI objectives. Each iteration provides a new functionality. This is accomplished via a constantly repeating pattern: "plan the iteration, commit to some functionality, execute the iteration by building and testing stories, demo the new functionality, hold a Retrospective, and repeat for the next iteration". At the end of each iteration, teams hold a system demo. The Innovation and Planning (IP) iteration occurs every PI. There, the teams have time for planning, and the opportunity for exploration, innovation, and retrospecting. The Big Picture in Figure 3.5 shows how a PI begins with a PI planning session and is then followed by four execution iterations, concluding with one innovation and planning iteration. There is no rule on how many iterations are in a PI, but experiences have shown that a PI duration of 8-12 weeks works best . Furthermore, an iteration is recommended to last two weeks, but can also last from one to four weeks.
The dev team is a subset of the agile team and is comprised of developers, testers, and various specialists.
The team level is part of the program level [80].

- Program level:
  The agile teams are divided into the ART, which is a virtual program structure. Each ART is a self-organizing, long-lived team of agile teams, consisting typically of 5 to 12 teams, along with other stakeholders, that plan, execute, commit, inspect, and adapt together. ARTs are organized around the enterprise's significant value streams and provide continuous objective evidence of progress, provide user experience and architectural guidance, facilitate flow, and align teams to a common mission. The ART produces releases or potentially shippable Increments at fixed time boundaries.
  At the program level, features and enablers that are required by the business to realize the vision and roadmap are defined and developed. To manage this, SAFe® provides a Program Kanban system, which is used to ensure that features are reasoned and analyzed prior to reaching the PI boundary and then are prioritized in the Program Backlog. Furthermore, it ensures that acceptance criteria have been established to guide a fidelity implementation.
  At the Program level, the Product Management (PM) role serves as the content authority for the ART and is responsible for identifying Program Backlog priorities.

In the Program Backlog, architectural and business features are defined and prioritized. Furthermore, the PM works with POs to optimize feature delivery and direct the work of POs at the team level.
A release train engineer (RTE) operates as the Chief Scrum Master for the train. The System Architect aligns with enterprise and solution architecture, and identifies and creates solution architecture to be delivered by teams' architecture.
The business owners are the role key managers that guide the ART to the appropriate outcomes.
The Weighted Shortest Job First (WSJF) is a formula that illustrates how the ART, the Solution, and the Portfolio Backlogs are reprioritized [80].

- Value Stream level:
  The Value Stream level, which is optional, supports the development of complex and large solutions, which require multiple, synchronized ARTs, as well as stronger focus on Solution Intent and Solution Context. Likewise, suppliers and additional stakeholders contribute to this level.
  The main purpose of this level is to describe lean-agile approaches that scale to the challenge of defining, building, and deploying large solutions. These require additional coordination, artifacts, and constructs. The Value Stream level also contains Program Increments, which are synchronized across all the ARTs in the value stream. By that, multiple ARTs, pre- and post-PI planning meetings, and the Solution Demo are synchronized.
  Additional roles at the value stream level include the solution management, the value Stream engineer, and the solution architect/engineering. The Pre-and Post Program Increment (PI) planning inform the ARTs of the Value Stream purpose and objectives [80].

- Portfolio level: The Portfolio level funds and organizes a set of value streams, which realize a set of solutions, to help the enterprise achieve its strategic mission. Furthermore, it provides funding for solution development by Lean-Agile budgeting, any necessary governance, and coordination of larger development initiatives that affect multiple value streams.
  This level has a bidirectional connection with business. One direction provides a constant flow of portfolio context back to the enterprise. The other direction provides the strategic topics that guide the portfolio toward larger and changing business objectives. This informs the enterprise of key performance indicators and other business factors affecting the portfolio as well as the current state of the solution set level.
  The Program Portfolio Management personnel have the highest level strategy and decision-making responsibility in SAFe®.
  The Epic Owner is responsible for conducting portfolio epics through the Portfolio Kanban system and developing the business case. When the epic is approved, he is working directly with the key stakeholders on the affected value streams to help actualize the implementation.
  Another role at the portfolio level is the Enterprise Architecture, who works with business stakeholders and Solution and System Architects to guide technology initiatives and drive enterprise standards across value streams [80].

The Foundation layer holds various additional elements, such as lean-agile leaders, communities of practice, core values, lean-agile mindset, and principles, which support development [80].

The Spanning Palette, which serves as a floating surface for roles and artifacts, is an essential part of the configurability and modularity of the Framework and can apply to multiple levels of SAFe®. The roles and artifacts belonging to the Spanning Palette include the System team, DevOps, Release Management, Shared Services, user experience, vision, roadmap, metrics, milestones and releases. These are also involved in the ART. These apply most often to the Program or Value Stream level. Some of them can apply to the Portfolio or Team levels (e.g. metrics, vision, roadmap, etc.) [80].

The four core values and nine principles of SAFe® provide the foundation for the framework. The values, namely alignment, built-in quality, transparency, and program execution, dictate behavior and action. These can help people know where to put their focus, distinguish what is right or wrong, and how to help companies to ascertain if they are on the right path to fulfill their business goals.
The principles that have evolved from agile principles and methods, Lean product development, systems thinking, and observation of successful enterprises determine the SAFe® practices.
The principles include:

- ”Take an economic view”;
- ”Apply systems thinking”;
- ”Assume variability; preserve options”;
- ”Build incrementally with fast, integrated learning cycles”;
- ”Base milestones on objective evaluation of working systems”;
- ”Visualize and limit WIP, reduce batch sizes, and manage queue lengths”;
- ”Apply cadence, synchronize with cross-domain planning”;
- ”Unlock the intrinsic motivation of knowledge workers”;
- ”Decentralize decision-making”.

The lean concepts used in SAFe® are based on the ”House of Lean” metaphor. The goal of lean is to deliver maximum value and quality to the customer in the shortest sustainable lead time [80].

Since the introduction of the SAFe®, a number of companies have applied the framework and published their experiences. Resulting benefits of these case studies include the increase in productivity of around 20-50%, faster time to market of around 30-75%, 50%+ reduction in deficit, and happier, more motivated employees [81].

Upon SAFe®, the Scaled Agile Inc. was built. Thirty five contributions were made and 38 case studies are reported. Documentation as well as communities, and training courses [89].

3.1.7.2. Differences between SAFe® 4.0 and 4.5

Recently, SAFe® version 4.5 was released. SAFe® 4.5 is backwards compatible with SAFe® 4.0, which means that organizations can adopt the new features in SAFe® 4.5 at their own pace. In contrast to the above described SAFe® 4.0, SAFe® 4.5 allows companies to test ideas more quickly using the Lean Startup Cycle and Lean User Experience (Lean UX), deliver more quickly with Scalable DevOps and the Continuous Delivery Pipeline, and simplify governance and improve portfolio performance with Lean Portfolio Management (LPM) and Lean Budgets. The new available implementation roadmap accelerates Lean-Agile transformation and guides enterprises every step of the way.
Furthermore, four configurable frameworks, which provide a more configurable and scalable approach, are introduced. These are Full SAFe, Portfolio SAFe, Large Solution SAFe, and Essential SAFe. The latter is the most basic configuration. It describes the most critical elements needed to realize the majority of the framework's benefits. The Large Solution SAFe® configuration is suitable for enterprises that are building large and complex solutions that require input of multiple ART and suppliers, but do not require portfolio considerations. Furthermore, it consists of the Large Solution level and Essential SAFe. The Portfolio SAFe configuration is suitable for enterprises that build solutions that also need to incorporate portfolio concerns. These may include strategy and investment funding, lean governance, and innovation across various value streams. Furthermore, this configuration adds the Portfolio Level to Essential SAFe. The most comprehensive configuration is Full SAFe. This one supports enterprises that build large, integrated solutions that require hundreds of people or more to develop and maintain.

Regarding the terminology, there are also some changes. The Value Stream Level in SAFe® 4.0 changed to Large Solution level. In addition, the terms 'Value Stream Backlog', 'Value Stream Engineer', 'Value Stream Epic', 'Value Stream Kanban' and 'Value Stream PI Objectives' changed to 'Solution Backlog', 'Solution Train Engineer', 'Solution Epics', 'Solution Kanban' and 'Solution PI Objectives', respectively.
Furthermore, a new term is introduced in SAFe® 4.5, namely Solution Train. This describes the organizational construct of SAFe®, which is used to build large and complex solutions that need to coordinate multiple ARTs, as well as the contributions of suppliers. In addition, it aligns ARTs and suppliers around a shared business and technology mission by using a common Solution vision, backlog, roadmap, and an aligned Program Increment (PI) cadence. The Solution Train Engineer is the leader of the train and helps the Solution Train run smoothly [75].

3.1.8. Disciplined Agile 2.0

The Disciplined Agile Delivery (DAD) is a process decision framework for lean enterprise [11] and was published in 2012 by Scott Ambler and Mark Lines [14]. The current version of the framework, named Disciplined Agile 2.0, was released in 2015. It provides lightweight guidance to help organizations streamline their information technology processes in a context-sensitive manner, by showing how various activities such as solution delivery, operations, enterprise architecture, portfolio management, and many others work together in a cohesive whole. The DA framework adopts practices and strategies from existing sources, such as Scrum, Kanban, XP and Agile Modeling, and provides advice for

when and how to apply them together [15]. The aim of DA 2.0 is to address areas that are not thoroughly covered in smaller scale agile frameworks and recommends three phases, namely inception, construction, and transition. It provides recommendations for processes that come both earlier in the project (inception) and as teams prepare for delivery (transition), while many other agile frameworks address what DA 2.0 calls the construction phase. Therefore, the strength of DA 2.0 is in providing more guidance in the areas of DevOps (transition), design, and architecture (inception).

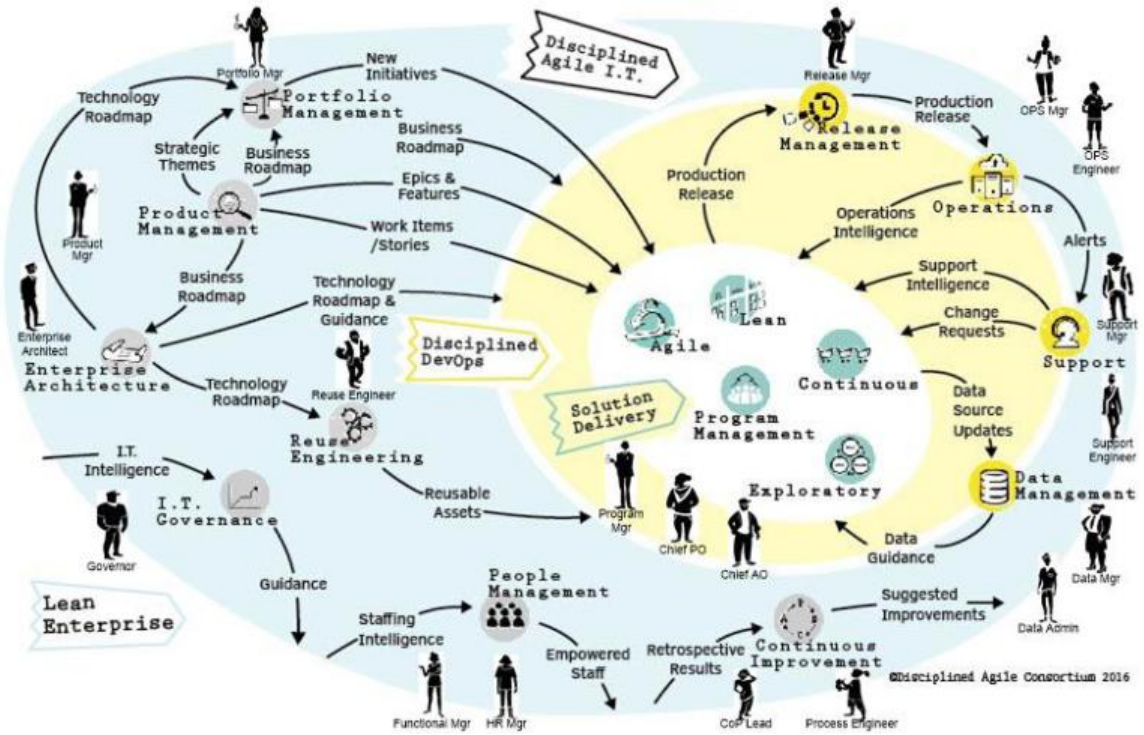


Figure 3.6.: The Disciplined Agile Delivery's Agile lifecycle [12]

Furthermore, it provides flexibility by suggesting different process guidelines for four categories of life cycles, namely lean/advanced, agile/basic, continuous delivery, and exploratory. The lean/advanced life cycle uses processes similar to Kanban, whereas the construction phase of agile/basic is Scrum. The inception phase aims to stock a work item pool that is organized to achieve business values, expedited delivery, fixed delivery dates, or some other goal. During the transition phase, planning, prototyping, stand up meetings, retrospection, and further activities are performed. The focus of the continuous delivery life cycle is on continuous integration, mature DevOps, and deployment processes for projects that require frequent delivery to stakeholders. The exploratory life cycle minimizes early planning due to fast delivery, gaining feedback and incorporating it into the next delivery. The inception phase in the continuous delivery life cycle is unique and has a brief transition period. In this life cycle, products are produced daily, weekly, or monthly. Last but not least, the exploratory life cycle aims to encourage agile teams to put themselves in start-up or research situations wherein the stakeholders have clear ideas for

a new product but do not yet understand the needs of their user base. The DA 2.0 roles, which are adopted from the Scrum and agile modeling (AM) methods, start with the PO and primary team members. The role of a team lead was adopted from the AM method. This role is similar to a Scrum Master or an architecture owner role. In order to address scaling issues, DA 2.0 has secondary roles, namely specialist, domain expert, technical expert, independent tester, and integrator [14]. In DAD, there are two types of "agility at scale" [11]:

- Tactical agility at scale is the application of agile and lean strategies on individual DAD teams. It aims to apply agile deeply to address all of the complexities, what are called scaling factors. The scaling factors include the ability to apply agile on teams of all sizes, on teams that are geographically distributed, on teams facing regulatory compliance, on teams addressing a complex domain, on teams applying complex technologies, on teams where outsourcing may be involved, and combinations thereof.

- Strategic agility at scale is the application of agile and lean strategies broadly across an entire organization. From an enterprise point of view this includes all divisions and teams within an organization, not just the IT department, whereas from an IT point of view, this includes Disciplined Agile IT in general and Disciplined DevOps.

The Disciplined Agile Consortium is built upon DA 2.0. Thirty four contributions were made and four enterprises use DAD. Documentation as well as communities, and training courses are available [89].

3.1.9. Spotify

Spotify is a model published in 2012 by Henrik Kniberg, Anders Ivarsson and Joakim Sundén [89]. The basic unit of development at Spotify is a Squad, which is similar to a team in Scrum [51]. Each Squad consists of 6-12 members [6].The teams have all the skills and tools needed to design, develop, test, and release to production. So far, Spotify has been scaled to over 30 teams. They are self-organizing and decide their own way of working — some use Kanban, some use Scrum Sprints, and some use a mix of these approaches. Each squad is encouraged to spend 10% of their time on "hack days" to promote learning and innovation. During hack days, people typically try out new ideas and share them with their buddies. Every Squad has a Product Owner who is responsible for prioritizing the work to be done by the team, but is not involved with how they do their work. In addition, a squad has access to an agile coach who helps them evolve and improve their way of working. A collection of squads that work in related areas is called a tribe. For every tribe, a tribe lead is assigned who is responsible for providing the best possible habitat for the squads within that tribe. Tribes regularly hold an informal get-together where they show the rest of the tribe what they are working on, what they have delivered, and what others can learn from what they are currently doing. The scrum of scrums happens "on demand".

At Spotify, there is a separate operations team, whose job is to give the squads the support they need to release code themselves. The chapter is a small group of people having similar skills and working within the same general competency area, within the same tribe.

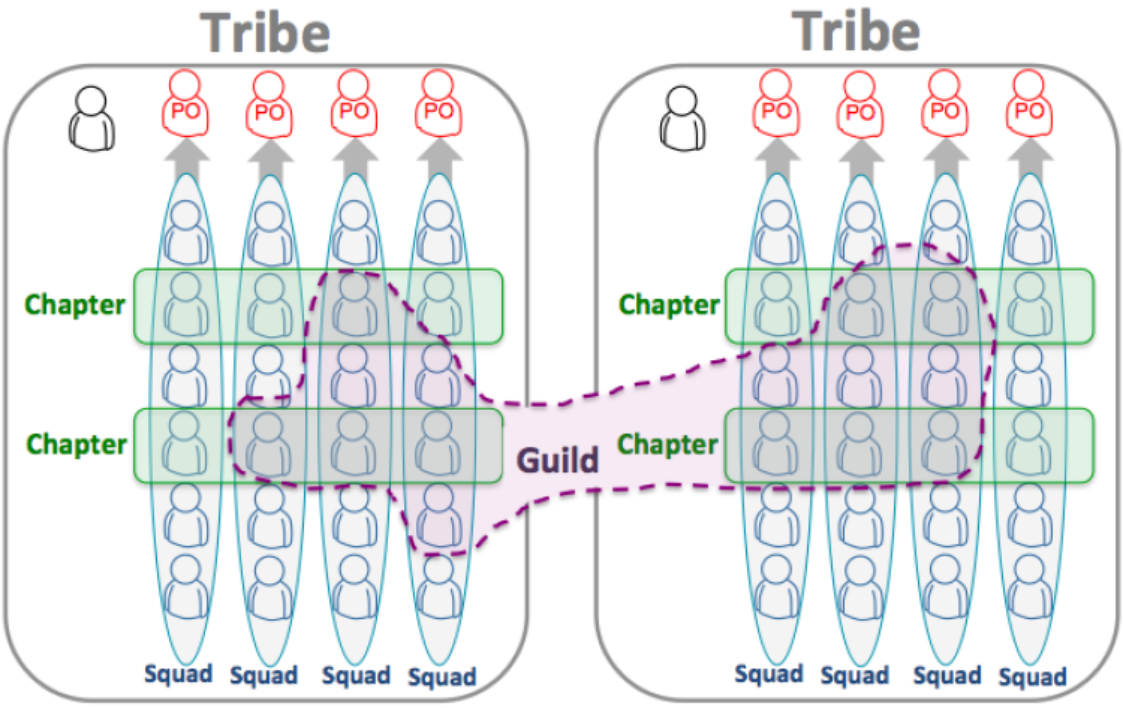


Figure 3.7.: The Spotify Model [51]

Each chapter meets regularly to discuss their specific challenges and their area of expertise. The line manager of the chapter members is the chapter lead. His responsibilities comprise developing people, setting salaries, etc. The chapter lead is also part of a squad and is involved in the day-to-day work. A guild is a group of people that want to share knowledge, code, tools, and practices. A guild usually cuts across the whole organization, while chapters are always local to a tribe [51].
The Spotify organization is built upon the Spotify model. Sixteen contributions were made, and one case study is reported. There are documentation and communities available, but no training courses [89].

3.1.10. Mega Framework

The Mega framework is a set of practices [83] and was published in 2012 by Rafael Maranzato, Marden Neubert and Paula Heculano [89]. It scales standard Scrum by allowing up to ten teams to work in parallel [48] and adding new meetings. The teams are feature teams instead of software component teams, and they work in four-week Sprints [83]. A feature team is a long-lived, cross-component, cross-functional team that completes many end-to-end customer features. Each team has its own development environment. After each release, the feature teams merge their code. In addition to the feature teams, the Mega Framework introduces a Mega Backlog, which is an initial series of discussions with the main stakeholders to organize the backlog. Besides the Scrum routine, the following meetings are added to the Mega framework. The Mega Planning happens after the Sprint Planning. It gives to the feature teams the opportunity to explain their stories to each other,

to know what is going on with other teams, and to receive feedback on how they intend to implement their stories. The Mega Stand-up happens after the middle of the Sprint and synchronizes the teams that are in the release. The goal of the Mega Retrospective is to find impediments that cross at least two teams but were previously considered less important or too difficult to deal with from a single team perspective. The difference between the Sprint Review in Scrum and the Mega framework is the audience. Besides inviting the team, the Product Owner, some members of the operational area, some of the stakeholders, and one representative of each other feature team is also in attendance. In the Weekly Pre-Planning meeting, only the features and stories that are candidates for the next Sprint are discussed. The most important Mega Framework meeting is the weekly Product Owner and Scrum Master meeting.

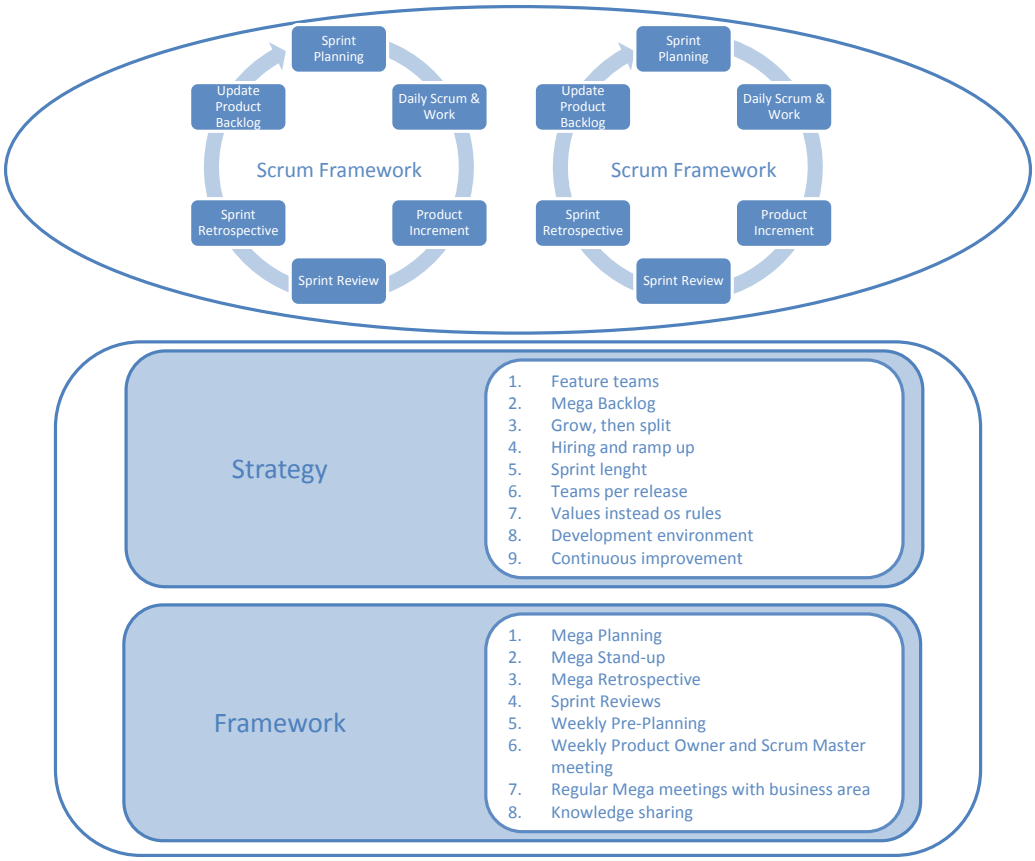


Figure 3.8.: The Mega Framework, Source: based on [83]

Here, the topics discussed include impediments, pending issues, corporate policies, decisions, and main features that are being developed or planned, especially those that can affect other teams; changes in the backlog planned for the next Sprints are also discussed. In the monthly Mega meeting with Business Area, pending issues, problems, and new opportunities for the product are discussed. As more people are added to the teams, it gets harder to reinforce and exchange good practices and knowledge. To facilitate this, the knowledge sharing meeting, which takes place once a week, is conducted for developers,

webmasters and test analysts [83].

The Universo Online S.A. is built upon the Mega Framework. Four contributions were made, and one enterprise uses this framework, but neither documentation, communities, nor training courses exist [89].

3.1.11. Event-Driven Governance

The Event-Driven Governance is a framework published in 2012 by Erik Marks [89]. Event-Driven Governance is light weight, lean, and virtual, supported by ten principles, processes, and designs. Trough self-governance and community-governance concepts, which help create a scalable, more collaborative, and trust-based approach to governance, it has become the first framework for enterprise agile development. The framework includes multiple agile teams that are protected through an agile governance buffer zone. Each agile team consists of a Scrum Master, a Product Owner, and the scrum team. The agile governance buffer zone protects agile teams from the friction of IT governance. Nevertheless, it enables them to engage with IT governance for escalations, critical decisions, and oversight in a way that is in and of itself agile. Furthermore, it improves governance process performance of agile delivery and allows agile teams to access oversight processes and enterprise governance, and align with IT Governance, compliance, and risk. The agile governance buffer zone describes a three tier governance, each of which describes a different type of IT governance. The three tiers include top-down prescriptive governance, community governance, and self-governance. In the top-down prescriptive governance, which is the traditional enterprise IT governance approach, policies are defined and enforced at the enterprise level. But this governance does not scale well, and is typically slow and unresponsive. Community governance style is based on a team, or a collective distributed team, to provide governance and oversight. It responds to governance requirements quickly, and also scales very well, because the community-based governance designs are simple, very responsive, and do not have the overhead of Top-Down prescriptive governance processes. In the self-governance, teams are empowered to self-govern based on the knowledge of key enterprise principles, policies, and controls. Moreover, it scales remarkably well. The Product Backlog is a central artifact in agile development, which guides the process and contains the user stories necessary for implementation of a product or a feature. Relating to enterprise IT governance, a new artifact is introduced: the governance backlog, which is similar to the Product Backlog. It contains prioritized governance requirements that are identified during different phases of the agile delivery process. These include the product planning, release planning, backlog grooming and Sprint planning. Furthermore, a new role is introduced: the agile governance owner, who is responsible for the governance backlog [84].

Upon Enterprise-Decision Governance the AgilePath organization is built. One contribution is made but no information about enterprises using EADAGP is available. There are documents and communities available, but no training courses [89].

3.1.12. Recipes for Agile Governance in the Enterprise

Recipes for Agile Governance in the Enterprise (RAGE) is a framework that was published in 2013 by Kevin Thompson [89]. It describes how governance can be conducted effectively

for Agile and hybrid processes in a large company that develops software applications.

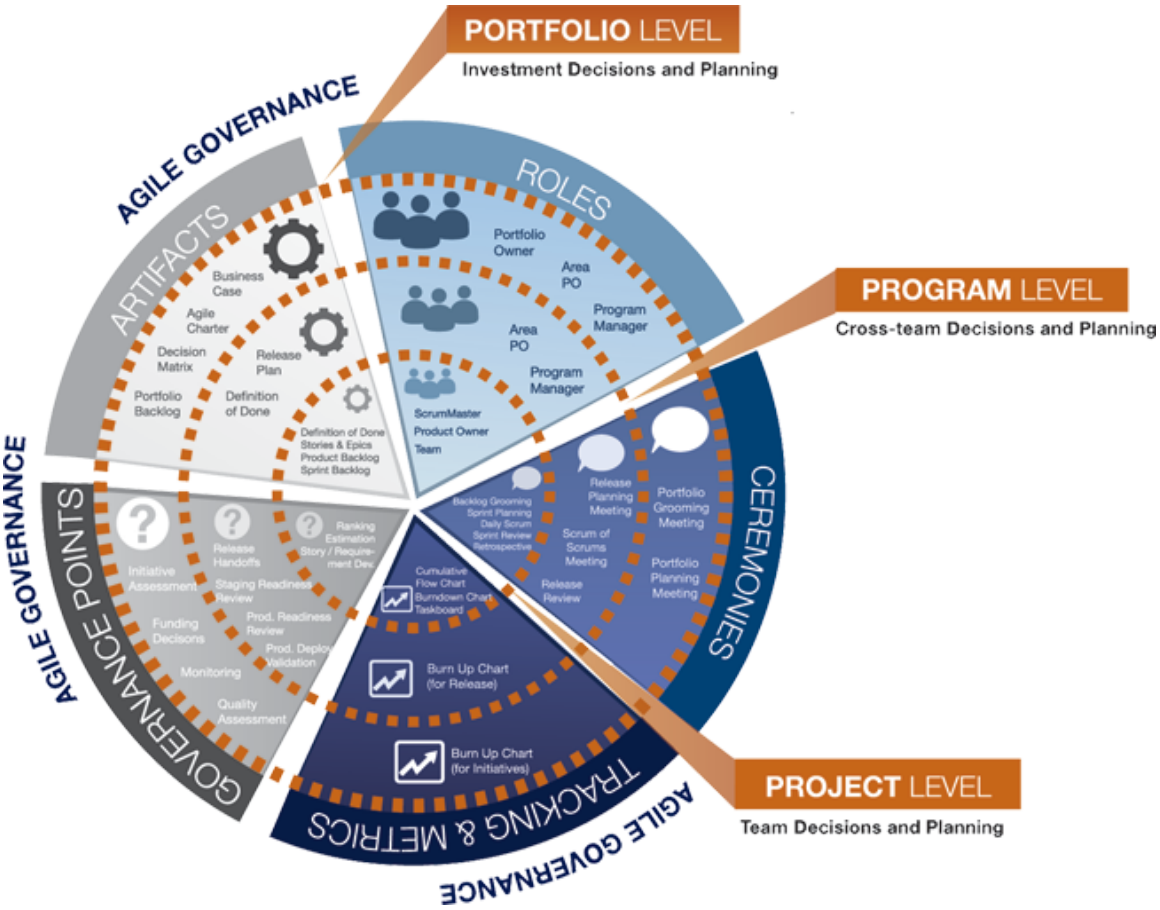


Figure 3.9.: Recipes for Agile Governance in the Enterprise [122]

The key insight is that the bulk of governance can be achieved by defining and using standard roles, meetings, artifacts, tracking and metrics, and governance points at different levels. The governance is divided into Portfolio, Program, and Project levels. For each level, appropriate practices do exist. The Project level refers to the work of a single team; the Program level refers to the collaboration between teams; and the Portfolio level refers to the development and management of business initiatives that lead to program- and project-level work. The roles at the Project level comprise the Product Owner, the Area Product Owner and the Program Manager. Meetings presented at the project level include the Portfolio Grooming meeting and the Portfolio Planning meeting. The Artifacts presented at the Portfolio level include The Business Case, the Agile Charter, the Decision Matrix and the Portfolio Backlog. The Program level distinguishes between Governance for Development in the Business Units and Governance for Deployment across the Business Units. An Area Product Owner and a Product Manager exist at the Program-level governance within Business Units, too. Existing meetings include the Release Planning meeting, the Scrum of Scrums meeting and the Release Review. The artifacts include the Definition of Done for the release and the release plan. The Program level governance across Business

Units includes the roles for the Release Manager, the Product Manager, and the Product Operations Team. Here, Release Hand, Staging Readiness Review, Production Readiness Review, Production Stand-up, and Production Deployment Validation meetings are held. At the Project level, the Scrum process and the Kanban process are available. The Scrum process is suitable for environments where there is need to plan the delivery of results on a schedule, but where it is possible to start and complete a set of tested deliverables within one to a few weeks, and where scope and effort are not well understood. Furthermore, it is suitable for Data Warehouse, Business Intelligence and software development, and other environments with similar characteristics. The roles in the Scrum process are the same as in standard Scrum. Besides the known Sprint Planning, Daily Scrum, Sprint Review and Retrospective meetings, the Scrum process additionally contains a Backlog Grooming meeting. Artifacts comprise the Sprint Backlog, the Product Backlog, the Definition of Done, and Stories and Epics. The Kanban process is suitable for environments where forecasting the delivery of results on a schedule is either not needed or impossible. It focuses primarily on prioritizing requests to do various kinds of work, which arrive at unpredictable moments, and organizing the workflow to optimize throughput. Furthermore, it is suitable for IT Operations, Production Support, and other environments with similar characteristics. In a Kanban process, requests for work to be done enter in the Backlog. Although Kanban does not define roles, RAGE proposes a Backlog Owner, a Process Master and the Team as useful roles. In the Kanban process, the Backlog Grooming, the Daily Stand-up and the Retrospective meetings are held. As Kanban has no prescribed artifacts, organizations whose Kanban processes move deliverables through workflow states often use the Story format as the specification of a deliverable [121].

The Cprime organization is built upon the RAGE framework. Six contributions are made, and one enterprise uses this framework. There are documentation, and training courses available, but no community, forum or blog [89].

3.1.13. Matrix of Services

The Matrix of Services (MAXOS) is a framework published in 2014 by Andy Singleton [89]. The MAXOS is used by organizations to scale the size, speed, and complexity of their IT operations and software development. The work to be done is prioritized in the Product Backlog. Software applications are divided into small services, each of which is built, tested, and operated by a small team. The service teams are small, typically consisting of 2-8 programmers who run the process and pull in other roles as needed. Thus, they can be fast and efficient, and they do not need to be multifunctional. The teams have their own integration test environment, and they test and submit changes via a distributed continuous delivery process. In MAXOS there is no iteration, which enables more scalability. The teams plan and communicate only if necessary, and integrate and release steadily. Continuous agile is continuous and non-blocking, lean, automated, and based on managing code. Consequently, the team members can work on their own features and do not have to wait for someone else to debug and release. This enables each person to work efficiently while, at the same time, enabling projects to scale to hundreds of contributors. Furthermore, contributors can concentrate on one task at a time while still doing a good job, and completing it. To manage the code, everything is put into repeatable scripts and is visible online. In this context, continuous agile uses source code management and code

contribution workflows [117].

In an e-mail, Singleton mentioned that SAFe is a bad idea [114] and that silicon valley companies would rather use the MAXOS structure and the "Apex strategies" [116].

Upon this framework the Maxos LLC organization is built. Five contributions were made but no information about enterprises using CAF is available. There are documentation and communities available, but no training courses.

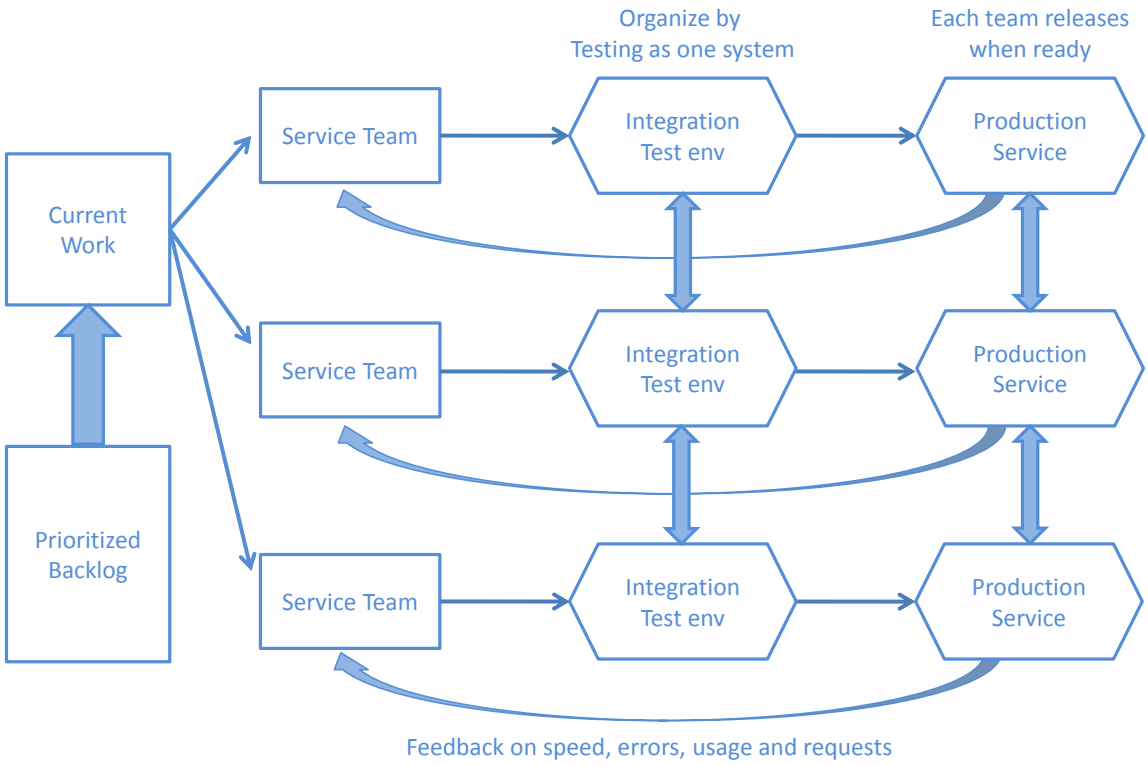


Figure 3.10.: Matrix of services, Source: based on [115]

3.1.14. Scrum at Scale

Scrum at Scale is a modular framework published in 2014 by Jeff Sutherland and Alex Brown [89]. It is a minimal extension of the core Scrum framework. The Product Ownership cycle generally describes the vision to complete a work, the process of how the teams will realize this vision, and the feedback from the customer. If the team visualizes how to complete the work, the work items have to be prioritized by importance. This is called the Backlog prioritization. Thereafter, this vision is broken down into smaller pieces. Over the time, the user stories need to be refined. This step is called the Backlog decomposition and refinement. During the Release Planning, the teams evaluate when to deliver a product. Each individual shippable increment is combined, finalized, and delivered to the customer. This is called the release management. The final step of the Product Ownership

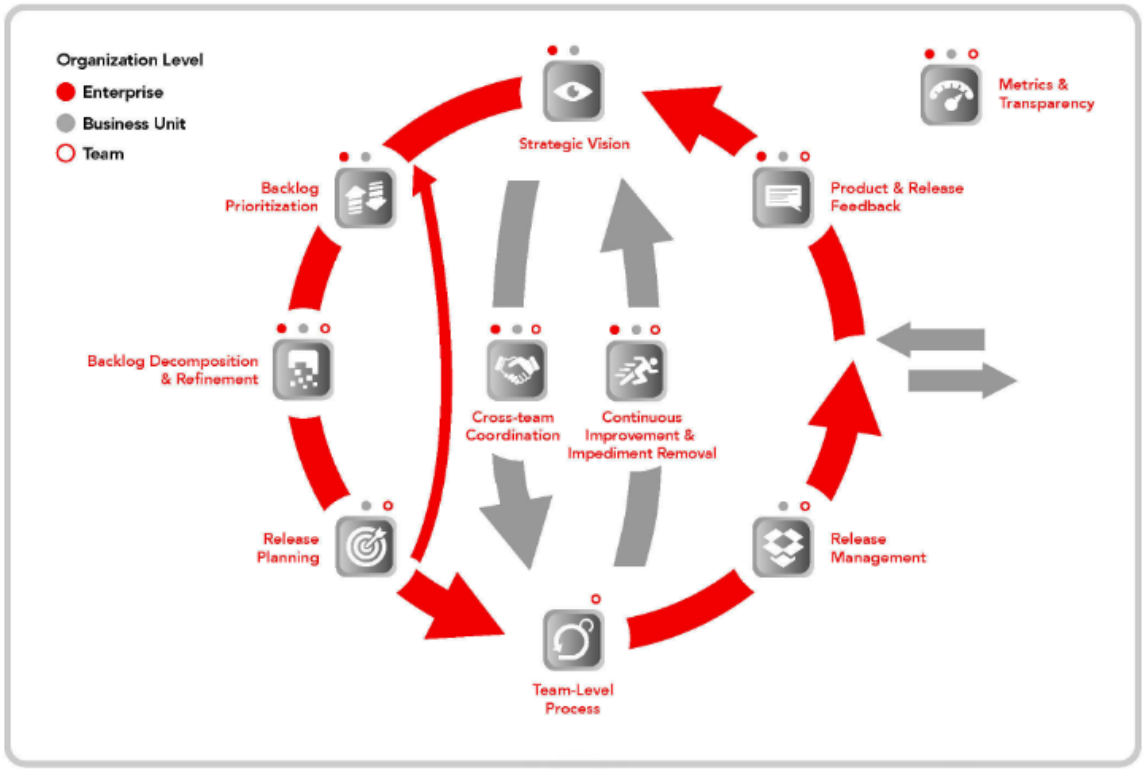


Figure 3.11.: Scrum at Scale [120]

cycle is the customer feedback. Based on the feedback, the vision is updated and refined. Another cycle included in the Scrum at Scale framework is the Scrum Master cycle, which starts at the team level process and boils up to the strategic vision. The idea behind that is continuous improvement and impediment removal. Based on the removal from impediments, a cross-team coordination, which is the closing Scrum Master review, is suggested additionally. The nine modular have to take place in a context of metrics and transparency. These concepts can be part of the team level, the product, project, or business unit level, or the enterprise level. Scrum at Scale allows scaling any Scrum implementation in a way that is tailored to the unique needs of a company without introducing anti-Scrum patterns or unnecessary waste. For example, Scrum at Scale helps to implement the Spotify model or improve the Scaled Agile Framework® implementation of an organization. In addition, it is also compatible with Large Scale Scrum and the Nexus Framework. The ability of context-driven solutions and processes is one of the keys to Scrum's success [5].

Upon this framework the Scrum Inc. is built. Thirteen contributions were made but no information about enterprises using this framework is available. Documentation as well as communities and training course and communities are available.

3.1.15. Enterprise Transition Framework

The Enterprise Transition Framework (ETF) was published in 2014 by agile42 [89]. It does not focus on specific methods like Kanban, SAFe®, or LeSS. Rather, it is based on fun-

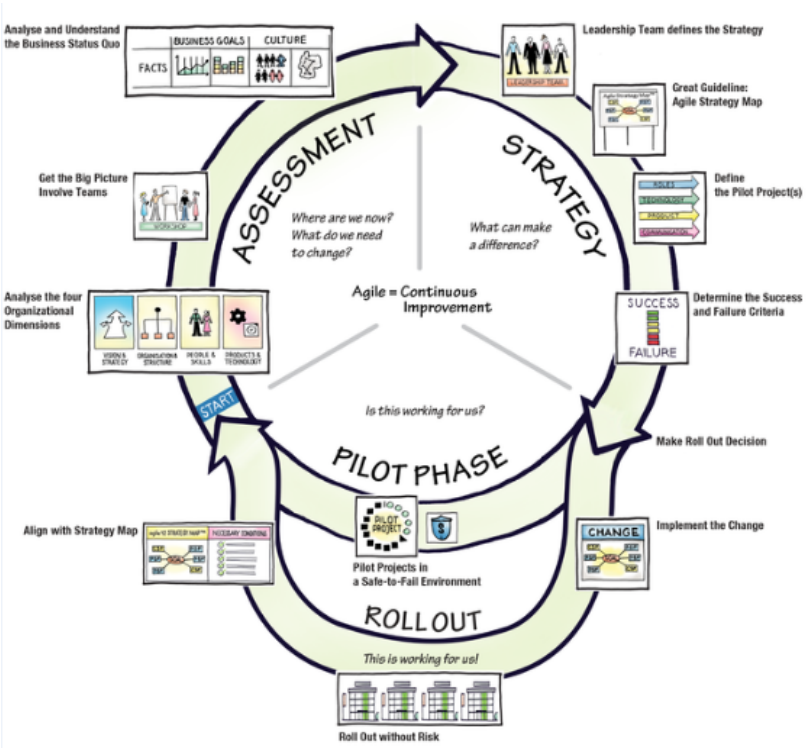


Figure 3.12.: The Enterprise Transition Framework [7]

damental agile principles and values. The agile42 organization has created the ETF that leads and supports any organization through the process of becoming more Agile. The focus of this framework is to allow an organization to implement continuous improvement and to experience change in a controlled way. The ETF supports the adoption of the Disciplined Agile Delivery (DAD), SAFe®, and other scaling agile frameworks. In addition, it enables structured and strategic organizational change, and will help to deploy SAFe®, DAD or any other framework. Furthermore, the ETF provides the right tools and methods to become agile, independent of the organization type. The ETF supports organizations in becoming resilient against market changes, and the Team Coaching Framework™guides agile coaches in their professional development [7].
To become agile, four steps are important [7]:

- ”Assessment: Understand the actual situation”
- ”Strategy: Identify the right agile strategies”
- ”Training: Create a common knowledge base”
- ”Coaching: Learn Agile principles for continuous improvement”

A set of useful methodical tools, coaching structures, and interactive training, such as the proven end-to-end approach of the ETF, can be scaled to organizations needs and size. The development of internal coaching capabilities is one important aspect of agile transition. For this the unique ”Team Coaching Framework (TCF)”, which contains also the education

of internal coaches, was developed. The education of internal coaches is called the "Coach the Coach" approach. The TCF speed up coaches' learning and helps organizations to effectively scale agile from the bottom-up as well as from the top-down [7].

Upon the ETF, the agile24 organization is built. Two contributions were made and two enterprises use this framework. Documentation as well as communities and training courses are available [89].

3.1.16. ScALeD Agile Lean Development

ScALeD Agile Lean Development (SALD) is a set of principles published in 2014 by Peter Beck, Markus Gärtner, Christoph Mathis, Stefan Roock and Andreas Schliep [89].

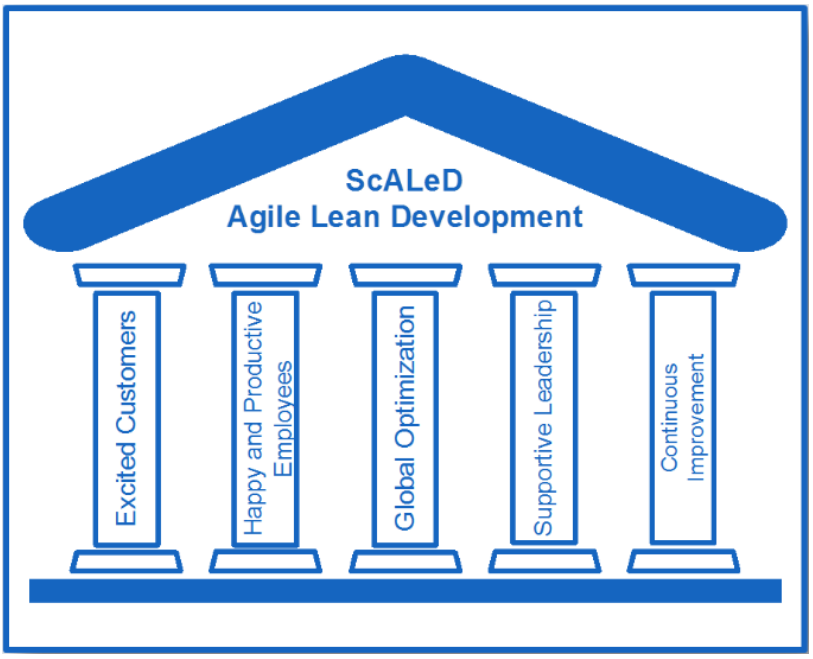


Figure 3.13.: ScALeD Agile Lean Development, Source: based on [18]

Believing that a set of guiding principles, rather than a new method or additional framework, is required to successfully apply agile practices at scale, Beck, Gärtner, Mathis Roock and Schliep defined a set of principles. The principles are not new, because they instead took existing ideas and reworded them to address the scalability challenges. Principles, such as excited customers, happy and productive employees, global optimization, supportive leadership and continuous leadership shall guide to applying agile at scale. Motivated customers enable businesses to maintain growth. A common understanding of the product's value proposition is especially important in a scaled organization. Shared values provide guidance for all project members. Small, deliverable increments are fundamental for the continuous growth of the product, because they minimize risks and reduce complexity. Employees offer the highest improvement potential, and achieve a higher productivity, if they are satisfied. Therefore, it is important to create a work environment that results in excited employees. Employees working on a large-scale agile project require not

only the right technical skills, but they also have to take on ownership for their work and support each other. Scaling requires a modular, loosely coupled product architecture. At all levels of the organization continuous improvement is an important agile practice. This is facilitated by repeated inspection and adaption. Inspection should be based on direct observation and communication, whereas adaption should happen without any delays [18].

Three contributions were made but no information about enterprises using SALD is available. There is documentation and communities available but no training courses [89].

3.1.17. Exponential Simple Continuous Autonomous Learning Ecosystem

Exponential Simple Continuous Autonomous Learning Ecosystem (XSCALE) is a set of principles published in 2014 by Peter Merel [89]. It is an acronym for the principles of an agile organization, as well as a language of best practice focused on exponential growth. The principles of the Agile Manifesto are extended to Portfolio Leadership, Product Leadership, Culture Leadership and Holarchy. The six principles are as follows [86]:

- ”eXponential return by stacking growth curves”;
- ”Simple design to the elegance of minimum”;
- ”Continuous optimization of throughput”;
- ”Autonomous teams, self-managing streams”;
- ”Learning: triple loop, breadth-first, set-based”;
- ”Ecosystems thinking: whole-board & win-win”

The Agile Manifesto focuses on agile teams, not organizations, which are composed of agile teams. Thus, they apply the agile values at all levels and to all business functions. In an XSCALE organization, the coach and leader serve as servant leaders. Organizations benefit from XSCALE, because Exponential Product Management aligns design, business and tech authorities breadth-first to optimize value stream throughput. In addition, De-scaling Patterns empower doers in decisions to prevent politics and disconnects. Furthermore, organization Permaculture minimizes costs of quality and delay, combining Agile with DevOps and ”open-book” Throughput Accounting. Finally, exponential transformation generates new capability without compromise or confusion [86].

Upon XSCALE the XSCALE Alliance is built. Five contributions were made, but no information about enterprises using XSCALE are available. Documentation as well as communities and training courses are available [89].

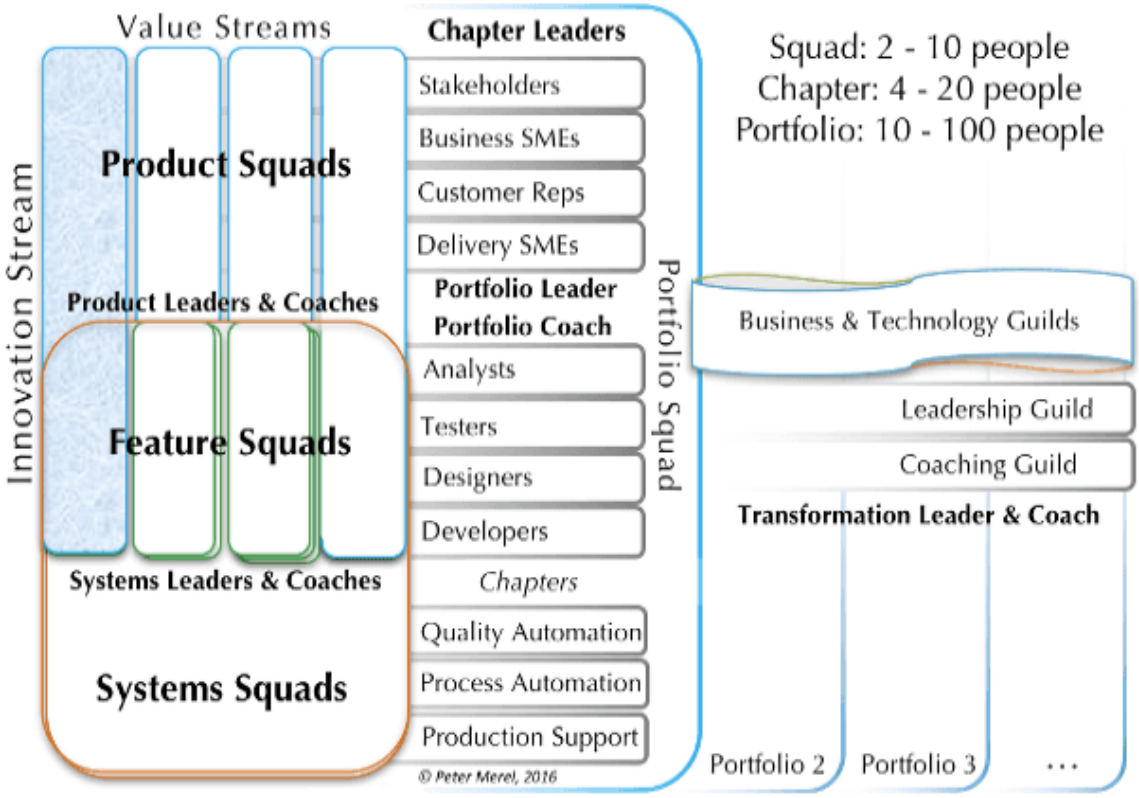


Figure 3.14.: The structure of Exponential Simple Continuous Autonomous Learning Ecosystem [87]

3.1.18. Lean Enterprise Agile Framework

Lean Enterprise Agile Framework (LEAF) is a framework published in 2015 by Satisha Venkataramaiah [126]. It is a tool of system thinking for helping organizational agility. In addition, it is a conversation starter that helps organizations to find constraints that limit their potential to scale the deliverables [127], rather than processes or teams [126]. An empirical process to design lean enterprises through kaizen culture is the essence of LEAF. The four pillars form the foundation of Leaf [127]:

- ”Systems Thinking”
- ”Engineering Practices”
- ”Community of Practices”
- ”Continuous Collaboration”

Upon this framework the LeanPitch Technologies is built. No contributions or information about enterprises using LEAF were found. Documentation as well as communities and training courses and certifications are available [89].

Unfortunaly, no further information or documentation was found regarding LEAF. The LeanPitch organization was contacted for additional information, but we did not receive any response.

3.1.19. Nexus

Nexus is a framework presented in 2015 by Ken Schwaber [89]. It is a framework for sustaining and developing scaled product and software development initiatives. Scrum is used as its building block. Compared to Scrum, Nexus pays more attention to dependencies and interoperation between Scrum Teams, delivering one "done" Integrated Increment at least every Sprint. Nexus consists of roles, events, artifacts, and techniques that

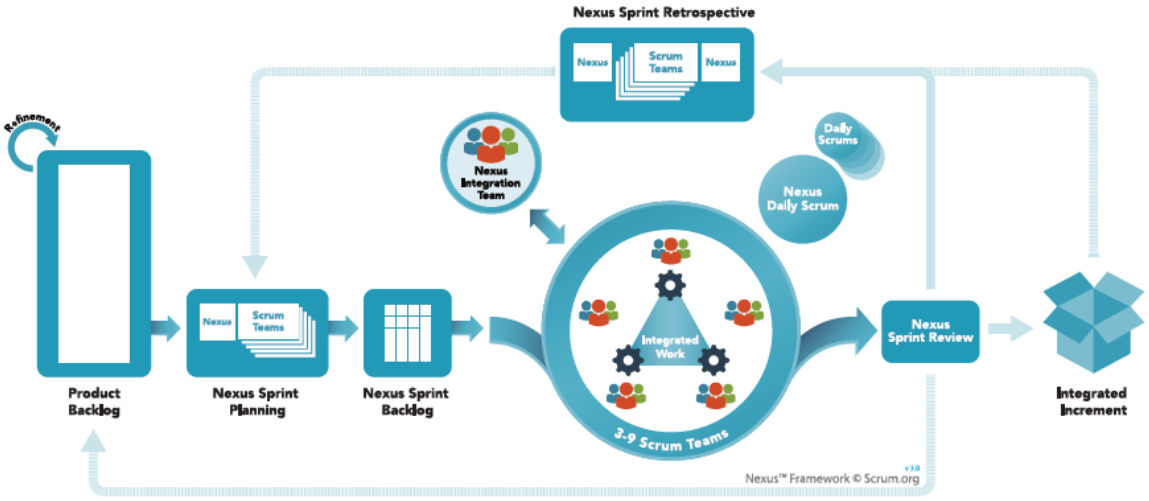


Figure 3.15.: The Nexus Framework [108]

bind together the work of approximately three to nine Scrum Teams working on a single Product Backlog to build an Integrated Increment that meets a goal. Most of these practices require automation, which helps to manage the volume and complexity of the work and artifacts especially in scaled environments. The Nexus Integration Team, a new role introduced in Nexus, exists to coordinate, coach, and supervise the application of Nexus and the operation of Scrum so the best outcomes are derived. It consists of a Scrum Master, a Product Owner, and team members. Similar to other methods, there exists a single Product Backlog, which is used by all Scrum teams and has a single Product Owner. Team members are responsible for coaching and guiding the Scrum Teams in a Nexus in order to acquire, implement, and learn these practices and tools. The Nexus Sprint Backlog, a new artifact, exists to assist with transparency during the Sprint. Each Scrum Team maintains their individual Sprint Backlogs. Nexus events comprise the Nexus Sprint Planning, the Nexus Daily Scrum, the Nexus Sprint Review and the Nexus Sprint Retrospective, where the latter consist of three parts. The first part enables representatives from across a Nexus to meet and identify issues that have impacted more than a single team. In the second part each Scrum Team holds their own Sprint Retrospective as described in the

Scrum framework. The third part is an opportunity for appropriate representatives from the Scrum Teams to meet again and agree on how to visualize and pursue the identified actions. There are many levels of refinement at the scale of Nexus. The Product Backlog items can be selected and worked on without conflict between Scrum Teams in a Nexus only when they are adequately independent. Product Backlog refinement at the enterprise scale forecasts which teams will deliver which Product Backlog items, and identifies dependencies across those teams. In doing so, it should focus on dependencies that have to be identified and visualized across teams and Sprints [108].

Upon this framework the Scrum.org organization is built. Six contributions were made but no information about enterprises using Nexus is available. Documentation as well as communities, and training courses are available [89].

3.1.20. Fast Agile

Fast Agile is a set of methods published in 2015 by Ron Quartel [89]. FAST Agile (FAST) is a system for collaboration from the small scale to the large. It combines elements of Scrum, XP, Open Space Technology (OST), and Story Mapping with the goal of high-quality and high-bandwidth output in short iterations. A FAST project may begin by merging all development teams working on the same project into one tribe. At the beginning, these teams stay together inside this tribe as clans. A Story Map forms the basis for the tribe's Product Backlog. There, all the features of the project are maintained and tracked visually. The entire tribe meets in cadence where each clan demonstrates the software that has been worked on since the last meeting. Afterwards, an Open Space style marketplace takes place where any team member may pull work from the backlog and announce what they will plan to do it in the next iteration. Thereafter, developers may either stay with their clan, or work with another clan. This is called flex-teaming. In the context of small scale, FAST can be used for scrum sized teams and in place of Scrum. Because FAST comprises parallel streams of work in an iteration, it works best if the work in progress size is at least 3. If the team is capable of working on three parallel streams then FAST can be used. In the context of large scale, a FAST tribe can maintain a size of around 150 developers. In case a project is larger than this, you have move to a multi-tribe mode. A tribe typically works on one project, but it is also possible for a tribe to own and work on multiple projects in parallel. In the case of restricted key resources, this is useful as the marketplace is a way to identify resource bottlenecks and dependencies and quickly strategize around this. In the multi-tribe mode, the separation of work between the tribes has to be clear. Each tribe will be responsible for one or more pieces of the portfolio. Product Owners meet to discuss the interrelationship of their products and all the implications and planning around this. This is similar to scrum of scrums, but includes the Product Owners. In addition, multi-tribe mode should be used for distributed teams. In this case, it is necessary that each distribution area becomes a tribe. That tribe takes ownership of one or more aspects of the project/portfolio. FAST requires a tribe to be co-located and to meet in person at each iteration. This is similar to the release planning meetings in the Scaled Agile Framework®, but happens each iteration. All of the current agile methods and scaling models have small cross-functional teams at their core. FAST is the first method, where these teams do not necessarily persist. However, FAST creates a cross-functional tribe, of which, teams self-form around the work. Within this tribe, any individual is allowed to move between teams

or form a team at any time. Furthermore, this tribe is a network and is a more efficient way of balancing work across teams and sharing knowledge than mechanisms used by other scaling models. Co-location is a precondition. Therefore, FAST differs from other scaling models because it is the only agile scaling model to state that co-location is a must. In FAST, there is no Scrum Master and no further administrative roles other than the Product Owner. FAST is founded on lean principles. By adding a Retrospective meeting into your FAST implementation and a Scrum Master role to the tribe, Scrum is implemented. A more structured way to express a Product Backlog is the story map, which is also iterative [99]. The Product Director role is similar to the Product Owner role in Scrum. FAST does not specify team leads or architect roles [100].

Upon the Fast Agile, the Cron Technologies is built. Three contributions were made, but no information about enterprises using FAST is available. There is documentation and communities available but no training courses or certifications [89].

3.2. Scaling Agile Frameworks in Research

In this section, related scientific works on scaling agile frameworks and their comparison are presented:

- Companies producing embedded systems are in the process of deploying agile methods, and several attempts to scale agile methods to include development of mass-produced systems can be identified. Some organizations developing mass-produced systems have successfully introduced agile development on the team level, but were facing challenges when doing so in largescale development of software intended for mass-produced systems. Eklund et al. [41] present these challenges. While some of them were described, SAFe® and DAD were mentioned in connection with the challenge of scaling the number of involved teams, as they involve release planning and road mapping of product portfolios. SAFe® is shortly described, whereas DAD is not described further.

- Alqudah and Razali [9] aim to review the existing literature of the utilized scaling agile methods by defining, discussing and comparing them. A wholesome understanding of the method, including its strengths and weaknesses as well as when and how it makes sense is important for large organizations to pick the right method. In a further analysis, SAFe®, LeSS, DAD, RAGE, Nexus and Spotify were considered to be scaling agile methods at large organizations. Therefore these frameworks are treated in Alqudah's and Razali's analysis. To give an understanding of these frameworks, each framework, including its roles and practices, is described. Subsequently, the frameworks are compared based on some criteria (See Table 4.1). Consequently, the results are summarized, and the advantages of each framework are shown.

- Ambler and Lines [10] state, that there are two fundamental visions about what it means to scale agile: tailoring agile strategies to address the scaling challenges and large team size. When scaling agile, agile teams can face scaling factors. These include team size, geographic distribution, organizational distribution, compliance, domain complexity and technical complexity. There are three features of the strategy

for scaling agile delivery strategies. The first feature is basic agile and lean methods. Methods such as Scrum, Extreme Programming, Kanban, and others are the source of principles, practices, and strategies that are the bricks from which a team will build its process. The second feature is the Disciple Agile Delivery (DAD), which provides an end-to-end approach for agile software delivery. The third feature is agility at scale. Teams which operate at scale apply DAD in a context-driven manner to address the scaling factors. In the context of agility at scale, you need to first scale agile delivery, before you can think about scaling agile across the organization.

- Brenner and Wunder [25] focuse on the applicability of the Scaled Agile Framework®. First, SAFe® 3.0 is described. Afterwards, a short case study of the AVL List company, where they adopted SAFe® 3.0, is described.

- Hayes et al. [44] discuss the dimensions of the scaling problem. The inventors of available frameworks that meet the scaling problem were interviewed. In this context, SAFe®, LeSS, DAD, DSDM and Scrum at Scale are described. This report prefers to describe each framework and provide graphics, references, and the advice of the author rather, than compare and contrast the frameworks for strengths and weaknesses.

- Kapadia [47] presents frameworks in context of multiple teams with total team members spanning tens or hundreds or even more. In this context, SAFe®, DAD and LeSS are mentioned. Primarily, the methodologist, the website and a short description for each framework are given. Afterwards, screenshots of each framework were shown, together with some bullet points. Finally, the three frameworks are compared based on some criteria (See Table 4.1).

- Korhonen and Luhtala [52] aim to contribute the understanding of prerequisites of implementing agile product development. As a framework that would support large-scale agile product development, Finland's Slot Machine Association decided to adopt SAFe®. As SAFe® plays a major role, the framework is described in detail. LeSS and DAD are only mentioned as other known scaling agile frameworks.

- The Global Teaming Model (GTM) recommendations, mentioned in [88], specify what a global software development project should do, but it does not specify how. In order to provide concrete guidance for projects that wish to employ agile methods in a global software development context, this paper aims to find out if the practices described in SAFe® could provide examples for how GTM recommendations could be actualized. Therefore, SAFe® is described in this Paper. Here, the ASK-Matrix is referenced to find other scaling agile frameworks.

- Ömer et al. [89] describe the roles of architects in scaling agile frameworks. A preliminary analysis of 20 identified scaling agile frameworks is shown. Subsequently, the three most popular scaling agile frameworks, namely the Scaled Agile Framework®, Large Scale Scrum, and Disciplined Agile 2.0, are described in detail. Finally, the roles of enterprise, software, solution, and information architects, as identified in six different scaling agile frameworks, are characterized.

- Pant [95] is about the Scaling Agile Frameworks and how they can be applied to any organization to get better business results. SAFe®, LeSS and DAD are compared based on six criteria (See Table 4.1). Based on this comparison, DAD proves to be an appealing option to make the agile transition painless. Therefore Discipline Agile Delivery is described extensively.

- Paasivaara and Lassenius [92] present a case study on the application of the LeSS framework to a large, distributed agile software project at Nokia. The project adopted Scrum immediately at the project start-up phase and used the LeSS framework to guide scaling. Primarily, SAFe® and LeSS are briefly described, and Disciplined Agile Delivery is defined. Besides these quite well-known frameworks, the Agile Scaling Knowledgebase Decision Matrix is referenced as documentation where further frameworks can be found. Afterwards, the case study is described. However, the project faced challenges in scaling Scrum, despite attempts at applying the LeSS framework. This is due to inherent problems in the framework itself and to the poor implementation.

- Large companies have been supported by Atlassian, as they adopt agile methodologies for many years. Atlassian provides a way to scale agile with the combined force of Portfolio for JIRA and JIRA Software. Radigan et al. [103] discusses how JIRA Software and Portfolio for JIRA can support the SAFe® methodology and organizational needs at all levels. After discussing the need for scaling agile, SAFe® 4.0 is described in detail. Finally, the tools used at each SAFe® level are presented.

- Razzak [104] describes the gaps of SAFe®. For example, SAFe® does not cover all aspects of agility required in a distributed environment context. Experienced adopters of SAFe® reported that geographically distributed teams experience lower productivity due to lack of alignment and solid program execution. Furthermore, SAFe® focuses only on describing the best practices, roles and artifacts of lean principles and agile but no attempt has been made to describe implementation strategy.

- Saeeda et al. [107] analyze agile approaches in detail, finding its success stories in small- and medium-sized projects, and highlighting its limitations for large-sized projects. This study identifies the current research problem of the agile scalability for large-size projects by giving a detailed literature review of the identified problem, and it synthesizes the existing work for covering the identified problem in the agile scalability. Two terms are introduced to distinguish between scalability at agile. Scaling up refers to using agile methods for developing large software systems that cannot be developed by a small team. Scaling refers to how agile methods can be introduced across large size projects with many years of software development experience.

- Stojanov and Turetken [119] develop a maturity model for adopting agile and SAFe® practices. The need to scale agile practices to large settings hides challenges. SAFe® is emerging as a key industry approach to address these challenges. Because well-structured approach for implementing and establishing SAFe® is lacking, organizations require a uniform model to establish a roadmap and to assess the current state and progress. In this context, SAFe® is described in detail.

- Turetken et al. [123] claim that there is a lack of a well-structured gradual approach for establishing SAFe®. Before and during SAFe® adoption, organizations can benefit from a uniform model for assessing the current progress and create a roadmap for the initiative. To address this need, a maturity model that provides guidance for software development organizations in defining a roadmap for adopting SAFe® is developed. In this context, SAFe® is described in detail. Furthermore, it shows the improvements and challenges that experienced organization faced after adopting SAFe®. As LeSS and DAD also belong to the most well-known scaling agile framework, they are described briefly.
- Vaidya [124] reviews DAD,SAFe® and LeSS and their approaches to roles, processes, and other salient features. This will provide some context on the practices they follow at Cambia as compared to those advocated by the scaling frameworks. The aim is to examine whether adoption of additional practices will lead to solve specific issues and make continuous improvements. To provide a frame of reference, they also compare and contrast the key features against Scrum.
- West et al. [136] represent a market guide for Enterprise Agile Frameworks. The identified frameworks are compared based on four criteria (See Table 4.1). In addition, brief descriptions of each provider and pointers to further information readily available in books, videos and websites are provided.

3.3. Limitations

The synthesized information sources provide a stable comprehension of scaling agile frameworks and their need in today's organizations. On the one hand, books and whitepapers provide deep insights into the respective scaling agile framework, whereas part of the scientific work contains detailed information about popular frameworks such as SAFe®, LeSS and DAD but does not even mention less common frameworks. However, the literature synthesis has revealed the following limitations:

- None of the analyzed scientific works name all existing scaling agile frameworks. They mostly enumerate some frameworks and give some details about the most important or rather best known ones. This limitation hampers enterprises in informing themselves about other frameworks if the mentioned ones are not suitable in their enterprise.
- None of the analyzed scientific works list all comparison criteria existing in other scientific works. Nor did any of them carry out an evaluation to check whether the mentioned comparison criteria are crucial for an enterprise to adopt a framework or are practicable at all.

These enumerated limitations unveil a research gap and corroborate the need for this thesis, creating a comparison template which enables enterprises to compare existing scaling agile frameworks and to decide which framework fits their enterprise best.

Part IV.

Comparison Table

4. Comparison Table

In order to create the comparison template, three steps are performed: Firstly, existing comparison criteria are gathered from identified publications. Nextly, the redundant criteria are eliminated. In total, 19 of 54 criteria were eliminated as they already exist in other publications. Furthermore, the criteria which do not seem to be important for an organization's decision to adopt a framework and criteria with a subjective content are eliminated, too. In this context, important ones include information regarding the structure, concepts, product, adoption, elements, and basic information of a framework. Criteria with a subjective content include the ones that can be interpreted differently and whose content is not clearly defined. In this step, further 22 criteria are eliminated. Finally, the remaining 13 criteria, together with ten self-created ones, were categorized. As a result, a comparison template with 23 criteria has been created, based on which every scaling agile framework can be compared.

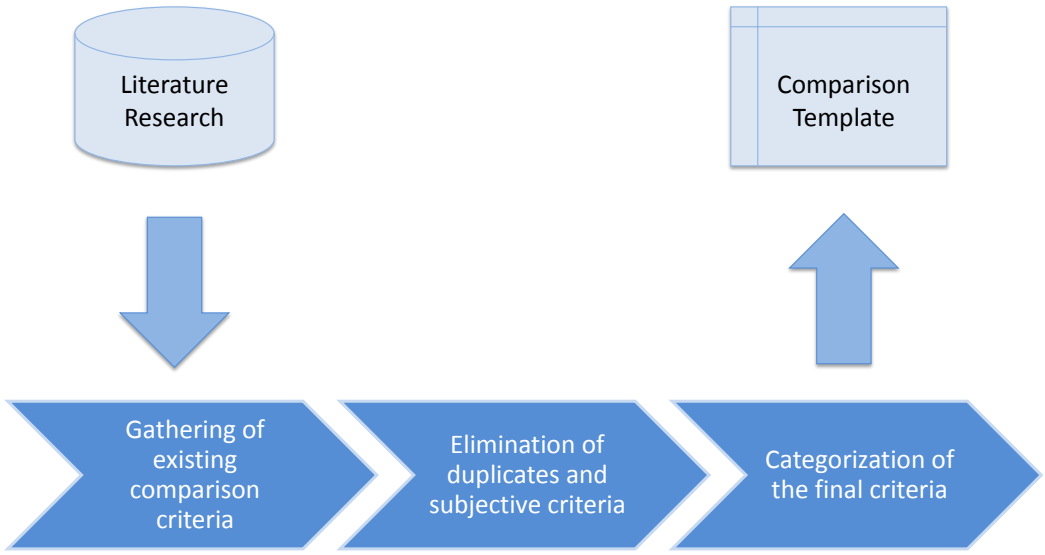


Figure 4.1.: Process of creating the comparison template

The following section describes the process of gathering, eliminating, and categorizing criteria, which leads to the resulting comparison template, in detail. Afterwards, the resulting criteria were described and compared.

4.1. Comparison Criteria

Out of the 148 identified publications about scaling agile development, only six sources were identified that compare scaling agile frameworks. From these six sources, 54 comparison criteria were identified. Scaling agile frameworks are compared by each publication as follows:

- Kapadia [47] compares the Scaled Agile Framework® (SAFe®), the Large Scale Scrum (LeSS), and the Disciplined Agile Delivery (DAD) based on *style, foundation, distinctive roles/teams, building block, distinctive events/meetings, websites, books*, and *certifications*.
  Here, relevant criteria are compared. It gives an overview of existing documentations and of existing components, such as roles and events.

- Alqudah and Razali [9] compare DAD, SAFe®, LeSS, Spotify, Nexus and Recipes for Agile Governance (RAGE) based on *team size, training and certificates, methods and practices adopted, technical practices required, organization type*, and *roles*.
  However, comparison criteria and the respective content provided by Alqudah and Razali are interesting, but the *technical practices required* criterion is filled out with ”low”, ”medium”, or ”high”. This assessment is too subjective, as we do not know what it really means or how it can be coded, i.e. under *technical practices required*, previous knowledge or the working period can be meant, but we do not know for sure.

- The ASK-Matrix [102] compares Scrum of Scrums, LeSS, SAFe®, DAD, Spotify, Drive Strategy Deliver More, RAGE, Nexus, and Scrum at Scale based on *description, web link, completeness of coverage of levels, portfolio, program structure, inter-team coordination, team level, tech practices, popularity/adoption, flexibility/emergence, typical cost to implement, availability of details and support, what team level frameworks are supported, emphasizes more central control or distributed, scale/target size, used typically by what organization types, focal point, software centric – how often used outside of SW or IT, big positives/key differentiators, key risks/concerns, training/resource availability*, and *deployment approach*.
  In comparison to other sources, [102] provides an extensive set of criteria, but some deficiencies in regarding subjectiveness, i.e. the assessment of the portfolio criterion based on three characteristics: ”low”, ”medium” and ”high”. However, there is no explanation of these three characteristics.

	Kapadia	Alqudah	ASK matrix	Pant	VanLeeuwen	West
style	✓					
methods and practices adopted	✓	✓	✓		✓	
roles	✓	✓				
building block	✓					
events/meetings	✓					
websites	✓		✓			✓
books	✓					
trainings and certifications	✓	✓	✓		✓	
team size		✓				
technical practices required		✓				
organization type		✓	✓			
description			✓	✓		✓
completeness of coverage of levels			✓			
portfolio			✓	✓		
program structure			✓	✓		
inter-team coordination			✓	✓		
team level			✓	✓		
tech practices			✓	✓		
popularity/adoption			✓			
flexibility/emergence			✓			
typical cost to implement			✓			
availability of details and support			✓			
emphasizes more central control or distributed			✓			
scale/target size			✓			
focal point			✓			
software centric– how often used outside of SW or IT			✓			
big positives/key differentiators			✓			
key risks/concerns			✓			
deployment approach			✓			
enterprise-targeted						✓
web-scale targeted						✓
methodologist						✓
source of practices						✓
published case studies					✓	✓
scientific information available					✓	

Table 4.1.: Existing comparison criteria in different sources

- Pant [95] compares DAD, LeSS, and SAFe® based on *description, portfolio, program structure, inter team coordination, team level*, and *tech practices*.
  The included criteria here are already described in [102]. Here, the content of the latter five criteria are also "low", "medium", or "high", without any explanation. Criteria like this are not meaningful and were considered to be too subjective.

- VanLeeuwen [65] compared the Agile Software Solution Framework (ASSF), DAD, Dynamic Systems Development Method (DSDM), Matrix of services (MAXOS) (Continuous agile framework), Enterprise Scrum (eScrum), LeSS, RAGE, SAFe®, the ScALeD Agile Lean Development (SALD), Scrum at Scale, Scrumban, Scrum of Scrums, the Spotify model, and the Mega framework based on *applicable on Scrum, (scientific) information available, relevant success cases, courses available*, and *agile*.
  Per se, the comparison criteria comprise interesting information, but the provided characteristics of the criteria are difficult to understand and are not meaningful. In case that a scaling agile frameworks matches the best with a criterion, it obtains ++. The scale from best to worst goes from ++ to +, o, -, –. This in turn represents subjective contents. Furthermore, the difference between *applicable on Scrum* and *agile* is not clear. In this thesis, we assume these two criteria as one.

- West compared AgilePath, Continuous Agile, Disciplined Agile (DA) 2.0, DSDM, FAST Agile, LeSS, Nexus, RAGE, SAFe® 4.0, SALD, Scrum at Scale, and Spotify based on the *methodologist, website, description, enterprise-targeted, web-scale-targeted, source of practices*, and *published case studies*.
  Thus, West [136] provides important basic information, such as the methodologist, the website, and a short description. However, the further criteria, such as enterprise-targeted and web-scale targeted, were not sufficient or meaningful, because we do not know exactly what it means and there is no explanation for it. Especially, the published case studies could not be accessed; it only indicates if there are some published or not.

4.2. Consolidation of Comparison Criteria

As one can see from the previous section, some of the aforementioned criteria are redundant, which are eliminated at the next step. These include all the criteria mentioned by Pant, as they are adopted from [102]. The criterion *description* in [136] is also present in [102]. In addition, the criteria *web link* in [102], and *website* in [136] correspond to the criterion *websites* from [47]. Furthermore, the criteria *distinctive roles/teams, certifications*, and *foundations* in [47] correspond to the criteria *roles, trainings and certifications*, and *methods and practices adopted* in [9], respectively. Furthermore, the criterion *building block* in [47] is renamed to *concepts* in our comparison template. The criteria *what team level frameworks are supported* in [102], *agile*, and *applicable on Srum* in [65] correspond also to the criterion *methods and practices adopted* in [9]. We renamed this criterion in our comparison template to *foundational practices and methods*. The criteria *courses available* in [65] and *training/resource availability* in [102] correspond to *training and certificates* in [9]. The criterion *relevant success cases* in [65] corresponds to *published case studies* in [136]. In our comparison template, this is reflected as the *number of stated cases*. Last but not least, the criterion *used typically by*

what organization type in [102] corresponds to the criterion *organization type* in [9]. In our comparison template, this criterion is renamed to *purpose*.

After eliminating these duplicates, we obtain 35 different comparison criteria. As there are still lots of comparison criteria, and some of them are not fundamental for the decision making of an organization or have a subjective content, they have to be verified and excluded, if applicable.
First, we eliminated the criteria *enterprise-targeted, web-scale-targeted*, and *source of practices* in [136], as well as *style* in [47], and *emphasizes more central control or distributed, focal point, Software centric – how often used outside of SW or IT, key risks/concerns, big positives/key differentiators* and *deployment approach* in [102]. The reason for this exclusion is that these are not considered to be decisive for an organization's decision.
Next, we eliminated *technical practices required* in [9], *completeness of coverage of levels, portfolio, program structure, inter-team coordination, team level, tech practices, popularity/adoption, flexibility/emergence, typical cost to implement*, and *availability of details and support* in [102]. The reason for this exclusion is that the characteristics are too superficially described, i.e., the portfolio criterion can be specified as either "low", "medium", or "high", but there is no explanation of what these three terms mean and no restriction to a value.
Furthermore, *scale/target size* in [102] is eliminated, as its content is too subjective.

After excluding all these criteria, only 13 criteria remain that are considered as important. Together with some other self-created criteria, we obtain 23 criteria in total.
During the consolidation phase, we have added the criteria *abbreviation, organization, publication date, category, availability of a community, scope level, approach for software development, iteration time period, artifacts*, and *purpose*. The first four criteria were added because these provide additional basic information concerning the regarded scaling agile framework. The availability of a community is added since it provides people with the possibility to exchange with other people and to always be up to date. The remaining criteria were added because based on those, the organization can build a more comprehensive opinion about the application and can better decide whether the regarded framework fits their organization.

Finally, the 23 criteria are grouped into three categories. Particularly, the background information of the scaling agile framework is categorized as *descriptive information*. The data, based on which organizations can search for proper readings and can exchange experiences, are classified as *adoption*. Finally, information describing the content of a framework is classified as *scope*. This results to the following comparison template in 4.2:

	Characteristic	Scaled Agile Framework	Large Scale Scrum
Descriptive information	Abbreviation		
	Short description		
	Methodologist		
	Organization		
	Publication date		
	Category		
	Website		
Adoption	Number of academic contribution		
	Number of stated cases		
	Documentation		
	Availability of training courses and certifications		
	Availability of a community/blog/forum		
Scope	Scope level		
	Foundational practices and methods		
	Purpose		
	Outcome		
	Approach for software development		
	Iteration time period		
	Team size		
	Roles		
	Events		
	Artifacts		
	Concepts		

Table 4.2.: Comparison template for scaling agile frameworks

The first category, *descriptive information*, comprises the *abbreviation*, a *short description*, the *methodologist*, the *organization*, which is built upon the framework, the *publication date*, the *category*, and the *website* for a scaling agile framework.
Many methodologists distinguish their way of scaling beyond the team level from a framework and rather call their method a *set of methods*, a *mechanism*, a *model*, a *set of principles*, or a *set of practices*. This distinction is represented in the comparison criteria *category*.

The second category, *adoption*, comprises the *number of academic contributions, number of stated cases, documentation, availability of training courses and certifications*, and the *availability of a community* of each scaling agile framework.
The *number of academic contributions* includes only scientific work; theses and presentations are not included.
The *number of stated cases* refers to the number of available case studies, and each is described on the homepage of the scaling agile framework under investigation.
Documentation includes available data in any form.
The *availability of training courses and certifications* includes those offered by each organization.
The *availability of a community* includes links to platforms of the scaling agile framework, where people can exchange with each other, and are kept up-to-date.

The third category, *scope*, comprises the *scope level, foundational methods and practices, pur-*

pose, *outcome*, *approach for software development*, *iteration time period*, *team size*, *roles*, *events*, *artifacts*, and *concepts* of each scaling agile framework.
The *scope level* describes up to which level of an organization the scaling agile framework can be applied.
Practices, on which the scaling agile framework is based, and methods, or some of their elements, used in the frameworks are present in *foundational methods and practices*.
The *purpose* describes with which types of organizations the framework would fit.
The *outcome* describes for which developing products the scaling agile is appropriate.
The *iteration time period* presents the timebox within which a potentially shippable product Increment is developed.
The *team size* comprises the number of teams and the number of members in each team.
Roles include people or teams that contribute to the development process, and support other people or the process.
Events include time-boxed meetings, each of which serves a purpose.
Artifacts describe work to be done or value that has to be delivered.
Concepts include elements that serve a purpose, and are not included in roles, events, or artifacts.

4.3. Comparison of SAFe® and LeSS

In this section, SAFe® and LeSS are compared based on the resulting comparison template in Section 4.2.

4.3.1. Comparison based on Descriptive Information

The Scaled Agile Framework®, also known as SAFe®, consists of proven patterns for implementing lean and ASD and system development at the enterprise scale [80]. It was first presented in 2011 by Dean Leffingwell. Upon this framework, the Scaled Agile Inc. was built [89].
The Large Scale Scrum, also known as LeSS, is one-team Scrum applied to many teams who work together on one product [62]. It was first presented in 2008 by Bas Vodde and Craig Larman. Upon the LeSS framework, the LeSS Company B.V. was built [89].

	Characteristic	Scaled Agile Framework	Large Scale Scrum
Descriptive information	Abbreviation	SAFe	LeSS
	Short description	SAFe is a knowledge base of proven patterns for implementing lean and agile software development and system development at enterprise scale	LeSS is one-team Scrum applied to many teams who are working together on one product
	Metholodogist	Dean Leffingwell	Craig Larman, Bass Vodde
	Organization	Scaled Agile Inc.	LeSS Company B.V.
	Publication date	2011	2008
	Category	framework	framework
	Website	www.scaledagileframework.com	www.less.works
Adoption			
Scope			

Table 4.3.: Comparison based on descriptive information

4.3.2. Comparison based on the Adoption

Regarding SAFe®, 35 contributions were made and 38 case studies are reported. Regarding LeSS, 29 contributions were made and 22 case studies are reported [89].

Documentation regarding SAFe® can be primarily found on the homepage [77]. Besides the details represented on the homepage, a whitepaper also exists [45], which summarizes and provides a good overview of SAFe®'s essence. Furthermore, six books [80, 138, 26, 50, 79, 78] were published.
Documentation regarding LeSS can be primarily found on the homepage [131]. Besides the details represented at the homepage, three books [62, 60, 57] about LeSS are available.

Both, SAFe® and LeSS offer trainings for specific roles, where people can obtain certifications if completed successfully. The certifications obtained through SAFe® include SAFe® 4 Program Consultant [66], SAFe® Agilist [67], SAFe® Release Train Engineer [73], SAFe® 4 Practitioner [71], SAFe® Scrum Master [74], SAFe® Advanced Scrum Master [68], and SAFe® 4 Product Owner/Product Manager [72].
Meanwhile, with LeSS, it is possible to obtain LeSS Practitioner, and LeSS for Executives [132].

On the SAFe® website there is a blog [69], where people are kept up to date concerning everything about SAFe®. For example, new available case studies or updates are an-

nounced. Furthermore, a community [70] for SAFe® exists where anybody can join and get the current news.
LeSS offers communities [129] and a blog [133] as well, where those interested in LeSS can discuss or ask questions.

	Characteristic	Scaled Agile Framework	Large Scale Scrum
Descriptive information			
Adoption	Number of academic contribution	35	29
	Number of stated cases	38	23
	Documentation	homepage, books, whitepaper	homepage, books
	Availability of training courses and certifications	yes	yes
	Availability of a community/blog/ forum	yes	yes
Scope			

Table 4.4.: Comparison of SAFe® and LeSS based on the adoption

4.3.3. Comparison based on the Scope

SAFe® and LeSS scale to all levels of an organization [80, 131]. In other words, they scale up to the enterprise and the Portfolio level [80].

SAFe® uses a combination of practices based on lean product development and flow, system thinking, and agile development [80]. Scrum and XP are foundational methods, which SAFe® combines and uses. Accordingly, SAFe® uses the project management practices of Scrum, and the technical practices of XP [80].
Similarly, LeSS uses a combination of practices based on various Agile and Lean concepts [124]. As the name indicates, LeSS is based on Scrum [62].

The ”3-level view” is well-suited for solutions that require a small number of agile teams, whereas the ”4-level view” supports those building large solutions that typically require hundreds or more practitioners to construct and maintain [80].
In comparison, LeSS is suitable for Large, Multisite and Offshore Product Development,

which includes inter alia embedded systems, telecommunications, or investment banking [56].

	Characteristic	Scaled Agile Framework	Large Scale Scrum
Descriptive information			
Adoption			
Scope	Scope level	all levels	all levels
	Foundational practices and methods	**practices:** lean product development and flow, system thinking, agile development **methods:** Scrum, XP, Kanban	**practices:** lean thinking, system thinking, agile development **methods:** Scrum
	Purpose	"**3-level view**" is well suited for solutions that require a modest number of Agile Teams "**4-level view**" supports those building large solutions that typically require hundreds or more practitioners to construct and maintain	Large, Multisite and Offshore Product Development
	Outcome	SAFe places an intense focus on working systems and resultant business outcomes **Value Stream level**: building large-scale, multidisciplinary software and cyber-physical systems **Portfolio Leve**l: the systems and solutions necessary to meet the strategic intent	broad complete end-to-end customer-centric solution
	Approach for software development	iterative and incremental	iterative and incremental
	Iteration time period	recommended as 2 weeks, but can last from one to four weeks	2-4 weeks
	team size	**Release Train**: 5-12 agile teams **Agile Team**: 5-9 persons	**Small LeSS**: 2-8 teams, with up to 8 team members **LeSS Huge**: more than 8 teams with up to a few thousand people

Table 4.5.: Comparison of SAFe® and LeSS based on the scope

SAFe® focuses intensively on working systems and resultant business outcomes. The application of the Value Stream level is suitable when building large-scale, multidisciplinary software, and cyber-physical systems. The Portfolio level ensures the development systems and solutions necessary to meet the strategic intent [80].
In contrast, LeSS is suitable for developing a broad complete end-to-end customer-centric solution that customers use. LeSS does not determine the scope of the product or the process of how to build it. Meanwhile, Vodde and Larman emphasize that organization should define their product broadly when applying LeSS, as they are more customer-centric [62].

Both frameworks use an iterative and incremental software development approach, as they deliver an Increment each iteration and integrate the resulted solutions. Moreover, bot frameworks have a timebox in which the product is revised and improved.

An iteration in SAFe® is recommended to last two weeks, but can last from one to four weeks [80].
Likewise, a Sprint in LeSS is recommended to last two to four weeks [63].

One ART in SAFe® includes 5 to 12 agile teams, each of which consists of 5-9 people,

making up the development team, the Scrum Master, and the Product Owner [80].
By contrast, Small LeSS is appropriate for 2-8 teams with up to 8 team members. If a project requires more than 8 teams, LeSS Huge has to be applied [62].

The roles in the Scaled Agile Framework® (SAFe®) at the Team level include [77]:

- the Product Owner — the content authority for the Team level. He is responsible for the Team Backlog, representing the customer to the agile team, and prioritizing and accepting stories;
- the Scrum Master — the servant leader and coach for an agile team. He educates the team in Scrum, XP, Kanban, and SAFe®. Furthermore, he ensures that the agile process is being followed;
- the Dev team — software developers and testers, engineers, and other dedicated specialists who can develop and test a story, feature, or component.

The roles at the Program level include [77]:

- the Release Train Engineer — the coach for the ART. He assists the teams in delivering value, and facilitates the events and processes;
- the System Architect/Engineer — represents an individual or small team and defines a common architectural and technical vision for the solution under development;
- the Product Management — responsible for the Program Backlog, for identifying customer needs, developing the program vision and roadmap, and prioritizing features;
- the Business Owners — a small group of stakeholders. They have the primary business and technical responsibility for compliance, governance, and return on investment for a solution, which is developed by an ART (ART).

The roles at the Value Stream level include [77]:

- the Value Stream Engineer — similar to the release train engineer. He facilitates and guides the work of all ARTs and suppliers. Both roles most often operate as leaders and have a good understanding of scaling lean and agile;
- the Solution Management — responsible for the Solution Backlog and works with customers to understand their needs, defines requirements, guides work through the Solution Kanban, and creates the solution vision and roadmap;
- the Solution Architect/Engineer — together with the system architect/engineer, helps bring into line the solution train and the ART to a common technological and architectural vision.

The roles at the Portfolio level include [77]:

- the Program Portfolio Management — the highest decision making responsibility within a SAFe® portfolio;

- the Enterprise Architect — drives strategic architectural initiatives for a SAFe® Portfolio, and facilitates adaptive design and engineering practices;
- the Epic Owner — coordinates portfolio epics through the Portfolio Kanban system.

The roles in the Spanning Palette include [77]:

- DevOps — builds the deployment pipeline. Furthermore, it facilitates automation and cooperation between operations and agile teams;
- the System Team — helps with infrastructure, assists with the System Demo and integration, performs ART-level testing, and is capable of evaluating conformance to nonfunctional requirements;
- the release management — responsible for guiding the value streams towards the business goals and help stakeholders to receive and deploy the new solution;
- Shared Services that supply the train with specialty functions, such as administrators, and business analysts;
- User experience designer — focuses on building systems. This reflects an understanding of end users, what they need and what they value, and an understanding of their abilities and limitations.

The roles in the Large Scale Scrum (LeSS) include [62]:

- one Product Owner — responsible for managing the Product Backlog and maximizing the value of the product, but at scale, rather focuses on keeping an overview and ensuring the maximum return on investment (ROI) in the product;
- the Scrum Master — teaches Scrum to the organization and helps them to reflect and improve towards their perfection vision. One Scrum Master is responsible for one to three teams;
- the Feature teams — cross-functional and cross-component teams that work together in a shared code environment, each doing everything to create done features every Sprint.
- the Area Product Owner — specializes in an area, and acts as Product Owner in relation to the teams for that area; does the same work as a Product Owner but with a more limited, and customer-centric perspective. This role is introduced in LeSS Huge.

The events in SAFe® at the Team level include [77]:

- the Iteration, wherein the teams built an Increment or product functionality. It is recommended to last 2 weeks, but can also last from one to four weeks;
- the Iteration Planning, where all team members specify how much of the Team Backlog they can commit during the next iteration. The Product Owner, Scrum Master, all the other team members, and any stakeholder are attending this meeting;

- the Team Demo, where the team reviews the Increment that results from the iteration. The agile team, ART stakeholders, business owner, customer, executive sponsors, and perhaps members of other teams are attending this meeting;
- the Iteration Retrospective, where team member discuss their work and identify ways to improve;
- the Daily Stand-Up meeting, where the teams coordinate their work. In this meeting, three question have to be answered by each team: "What stories did I work on yesterday (and their status)?", "What stories will I be able to complete today?", "What is getting in my way (am I blocked)?".

The events at the Program level include [77]:

- the System Demo, where the subject system being built by the ART is demonstrated. In addition, the system is tested and evaluated to get feedback from the primary stakeholders.
- the Program Increment (PI) planning, where business context and vision are presented. Thereafter, the teams create the plans for the upcoming PI.
- Inspect and Adapt, which is held at the end of each PI. There, the current state of the solution is demonstrated and evaluated. All program stakeholders attend this event. A similar inspect and adapt event is held for large solutions at the Value Stream level.

The events at the Value Stream level include [77]:

- the Pre- and Post PI planning, which supports and coordinates multiple ART involved in the value stream. This plannings allow the teams to build a plan for the next PI and occur just prior to, and just after the ART plannings. In the Pre-PI Planning meeting, the context for the upcoming ART PI Planning sessions are set. In the Post-PI Planning session, the results of ART planning are integrated into value stream objectives for the upcoming PI and into the solution roadmap;
- Solution Demo, where the results of all the development efforts from multiple ARTs are shown to the customers and other stakeholders.

The events in LeSS include [62]:

- the Sprint, in which a potentially shippable product Increment results. It is recommended to last 2-4 weeks;
- the Sprint Planning One, where all of the teams come together and decide which team will work on which items;
- the Sprint Planning Two, which is a separate meeting for each team. There, each creates the plan for getting the items to "done" during the Sprint.
- the Daily Scrum, which is the same as in one-team scrum. The Scrum of Scrums is another coordination technique related to the Daily Scrum and happens three times a week. There, representatives of each team get together and decide what coordination is needed across teams;

- the Sprint Review, where the teams review the one potentially shippable product Increment together. In addition, customers and stakeholders examine the increment of the teams and discuss changes and new ideas;
- the Sprint Retrospective, which occurs at the end of the Sprint. This is a separate meeting for each team in which a plan for improvements to be enacted during the next Sprint is created;
- the Overall Retrospective, which is held after the team Retrospectives. There, cross-team and system-wide issues are discussed, and improvement experiments are created. Furthermore, a retrospect on the previous Sprint from a product perspective takes place.

Scope	Roles	**Team level**: Product Owner, Scrum Master, Development team **Program level**: Product Manager, Release Train Engineer, System Architect, Business Owner, Stakeholder **Value Stream level:** Value Stream Engineer, Solution Manager, Solution Architect **Portfolio level:** Epic Owner, Enterprise Architect	**Small LeSS**: Product Owner, Scrum Master, Feature team **LeSS Huge**: Product Owner, Scrum Master, Feature team, Area Product Owner
	Events	Iteration Planning, Program Increment (PI) Planning, Daily Stand-Up, Iteration Retrospective, System Demo, Inspect and Adapt, Iteration Review, Pre- and Post-PI Planning, Solution Demo	Sprint, Sprint Planning One, Sprint Planning Two, Daily Scrum, Sprint Review, Sprint Retrospective, Overall Retrospective
	Artifacts	Team Backlog, Program Backlog, Portfolio Backlog, Value Stream Backlog	**Small LeSS**: Product Increment, Product Backlog, Sprint Backlog, **LeSS Huge:** Area Product Backlog
	Concepts	Program Increment, Agile Release Train, Value Streams, Architectural Runway, Spanning Palette	Requirement Area

Table 4.6.: Comparison based on the scope

The SAFe® artifacts include [77]:

- the Team Backlog, which contains user or enabler stories that a team needs to do to advance their part of the system;
- the Portfolio Backlog, which holds and prioritizes Epics that have been approved for implementation;

- the Program and Value Stream Backlog, which comprises the repositories for all the upcoming work necessary to advance the solution. The Backlog consists of pending features (Program Backlog) and capabilities (Value Stream Backlog). Features and capabilities deliver business benefits, and Enablers that advance learning and build the Architectural Runway. Furthermore, they address user needs;

The artifacts in the Spanning Palette include [77]:

- the vision, which describes the view of the solution to be developed, including customer and stakeholder needs, features, and capabilities;
- the roadmap, which transmits ART and value stream deliverables and includes the visibility into the deliverables of the current PI;
- the metrics, which are used to measure the systems and evaluate the performance;
- the milestones, which are marked progress points on the development timeline;
- the release, which is a valuable, working, and tested increment of the solution.

The LeSS artifacts include [62]:

- the product Increment, which is the resulting product of every Sprint;
- the Product Backlog, which defines all of the work to be done on the product;
- the Sprint Backlog, which contains the work that the team will need to do to complete the selected items of the Product Backlog;
- the Area Product Backlog, which is a view into the Product Backlog based on the requirement area. For every area Backlog, a Product Backlog item belongs, and the other way round.

SAFe® introduces new concepts. These include [77]:

- the ART, which is the primary value delivery construct. It is a team of agile teams and comprises 5-12 teams that plan, commit, and execute together.
- the Program Increment, which is a cadence-based interval for building and validating a full system increment, getting fast feedback, and demonstrating value. At the end of every Program Increment (PI), a System Demo and an Inspect and Adapt workshop occurs. The Program Increment is the same for an ART, as an iteration for an agile team is;
- the Innovation and Planning (IP) iterations, which gives teams the opportunity every PI to work on activities that are difficult to fit into a continuous, incremental value delivery process. During the IP iterations, teams have inter alia time for innovation and exploration, and the final integration of the solution.
- a Value Stream, which is a series of steps that an enterprise uses to deliver a continuous flow of value to a customer;

- the Spanning Palette, which serves as a floating surface for roles and artifacts. It is an essential part of the configurability and modularity of the Framework and can apply to multiple levels of SAFe®. It is most often applied to the Program level and Value Stream level, but some elements can also apply to team level and Portfolio level;
- the Architectural Runaway, which provides the necessary technical basis for implementing new features or capabilities, and developing business processes;
- the Solution Intent, which is also known as critical knowledge. It gives an understanding of the current and evolving requirements and design of the solution being built;
- the Solution Context, which identifies critical aspects of the target solution environment, and impacts features, capabilities, development priorities, and nonfunctional requirements.

LeSS Huge introduces one new concept: requirement areas. This provides structure if there are more than eight feature teams and complements the concepts behind feature teams. Furthermore, it is a categorization of the requirements, which lead to different views of the Product Backlog [62].

4.3.4. Commonalities and Differences

All in all, both frameworks introduce new concepts and offer several roles, artifacts, and events. With respect to the roles, it is obvious that SAFe® offers many more roles than LeSS does. Nevertheless, there are some commonalities concerning the roles, artifacts, and events.
The following section provides an overview about the commonalities and differences between these two practices.

The Product Owner in SAFe® is the owner of the Team Backlog, and the Product Owner in LeSS is the owner of the Product Backlog. The Team Backlog contains all the things a team needs to do to advance their part of the system. The Product Backlog defines all of the work to be done on the product. From this, two things can be referred: The Team Backlog in SAFe® corresponds with the Product Backlog in LeSS, and the Product Owner in SAFe® corresponds with the Product Owner in LeSS.
Furthermore, in both frameworks, the Scrum Master is the owner of processes and efficacy. The reason for this is that the Scrum Master ensures that the teams employ an agile method, and that their processes improve.
The timebox, in which the teams deliver a potentially shippable product, corresponds to the iteration in SAFe® and the Sprint in LeSS. The iteration in SAFe® is recommended as two weeks, and can last from one to four weeks, whereas the iteration in LeSS can last from two to four weeks.
At the end of each iteration, the teams review their increment that resulted within an iteration. In SAFe®, this happens in the team demo. In LeSS, this happens in the Sprint Review. These assumptions are also supported by Foegen [42], which describes customizable patterns for the adoption of scaling agile frameworks. In this context, [42] compares SAFe® and LeSS based on artifacts, events, and roles.

In SAFe®, the development team develops and tests the features each iteration, whereas in LeSS, the feature team delivers features every Sprint. This implies that the development team in SAFe® correspondents with the feature team in LeSS.
The Product Owner, Scrum Master, and development team in SAFe® together yield the agile team. The Product Owner, Scrum Master, and feature team in LeSS together yield the Scrum team.

At the beginning of each iteration, the teams create their plan for the pending iteration in a meeting. This happens in SAFe® in the Iteration Planning, whereas in LeSS, the teams select and plan in the Sprint Planning One and Sprint Planning Two, for which items are to be delivered at the end of each Iteration.

During each Iteration, a daily meeting is held. Therein, each team member has to answer what he worked on the past day, what he will do that day, and what impediments exist. In SAFe®, this is called the Daily Stand-Up, whereas in LeSS, this is called the Daily Scrum.

Furthermore, the work of the teams is discussed at the end of each iteration and ways of improvements to be enacted in the future are found. In SAFe®, this happens in the Iteration Retrospective meeting. In LeSS, this happens in the Sprint Retrospective meeting.

	SAFe	LeSS
artifacts	team backlog	product backlog
events	iteration	sprint
	iteration planning	sprint planning
	iteration retrospective	sprint retrospective
	daily stand-up	daily scrum
	team demo	sprint review
roles	development team	feature team
	product owner	product owner
	scrum master	scrum master
	agile team	scrum team

Table 4.7.: Common artifacts, events and roles in SAFe® and LeSS

- LeSS and SAFe® draw from a variety of agile and lean practices and use a combination of practices based on various agile and lean concepts. Both frameworks try to

address practices beyond the team level [124].

- SAFe® offers a more governance oriented framework with clear roles and responsibilities linking multiple development teams with the enterprise, whereas LeSS offers a more scalable process and goes into detail about the underlying principles of agility at scale [33].
- SAFe® assumes an intentional architecture and emergent design. That means that SAFe® provides guidance to ensure that the system has integrity and efficiacy, and through the emergent design, teams can appropriately react to changing requirements [80]. Furthermore, it covers the roles of enterprise, software, solution, and information architects [89]. LeSS assumes an emergent design [58], but does not cover the roles of architects [89]. The emergent design in LeSS is based on the observations of the paths people naturally walk, and to create permanent paths along these desire lines, as wide as appropriate [58].
- Small LeSS and the 3-level-view of SAFe® are both designed for few agile teams, whereas LeSS Huge and the 4-level-view in SAFe® support building large solutions that typically require hundreds or more people to build and maintain [80, 62].
- LeSS sticks very much to the traditional one-team Scrum, as it has nearly the same structure and adopted the terms for the roles, events, and artifacts presented in Scrum [62]. SAFe®, however, introduces a different structure and many more new conceptions [80].

Part V.

Conclusion

5. Conclusion and Outlook

This chapter summarizes the thesis in Section 5.1, highlights the key findings in Section 5.2, reveals the limitations of the thesis in Section 5.3, and provides a brief outlook of possible future investigations in Section 5.4.

5.1. Summary

The aim of the thesis was to create a comparison template based on which any scaling agile frameworks can be compared.
At the beginning of this thesis, the motivation for the raison d'être of scaling agile frameworks was described, followed by the deduced objectives and underlying research approach in order to ensure rigor and relevance.
To provide a better understanding of scaling agile, firstly, some background information about the traditional ASD and lean thinking were presented. In addition, Scrum and XP, as the two most popular ASD methods, and Kanban and Scrumban, which are based on lean thinking, were described extensively. Afterwards, the meaning of scaling agile and its difference to the traditional ASD was described.
In the context of scaling agile, all existing scaling agile frameworks were superficially described and hereby, the two most popular practices — SAFe® and LeSS — were described in detail.
Furthermore, existing comparisons about scaling agile frameworks in several publications were analyzed. Firstly, all comparison criteria were gathered. In the next step, duplicates, and not suitable cases were eliminated. Together with some other self-created criteria, we then categorized them. This resulted in a comparison template with 23 criteria with three respective categories. Based on this comparison template, SAFe® and LeSS were compared.

5.2. Results

Besides the aim of creating a comparison template, this thesis further aimed at comparing SAFe® and LeSS, in order to deduce commonalities and differences.
In order to create this comparison template, we first analyzed and identified publications about scaling agile frameworks. Out of the 148 sources, only six of them compare scaling agile frameworks. In total, 54 comparison criteria were identified. In the next step, redundant criteria were eliminated. After eliminating these duplicates, this resulted in 35 different comparison criteria. Thereafter, criteria with a subjective content were eliminated. After excluding all these criteria, out of the 35 criteria, 13 were considered to be important. Together with ten other self-created criteria, 23 criteria resulted into the comparison template.

Based on this comparison template, we compared SAFe® and LeSS. The comparison has shown that LeSS and SAFe® have several commonalities, although it does not look like it at first sight. SAFe® looks much more complex and extensive than LeSS does, because it offers many more roles and new concepts.
Vodde an Larman justify their decision not to introduce more roles or artifacts by saying [62]:

"We don't want more roles because more roles leads to less responsibility to Teams. We don't want more artifacts because more artifacts leads to a greater distance between Teams and customers. We don't want more process because that leads to less learning and team ownership of process. Instead we want more responsible Teams by having less roles, we want more customer-focused Teams building useful products by having less artifacts, we want more Team ownership of process and more meaningful work by having less defined processes. We want more with less."

But on the one hand, not every introduced role or concept is a must, and on the other hand, many roles, artifacts, and events are also present in LeSS. The comparison has shown that both frameworks are founded on ASD, lean thinking, and Scrum. The roles of the Product Owner, Scrum Master, and the development team in Scrum is recognized in both SAFe® and LeSS. Furthermore, the events in Scrum, such as the Sprint Planning, the Daily Scrum, Sprint Retrospective, and Sprint Review are also represented in SAFe® and LeSS. The Product Backlog in Scrum is also present in SAFe® and LeSS.
Here, it is noticeable that LeSS sticks very much to the traditional one-team Scrum, as it has the same structure and adopted the terms for the roles, events, and artifacts presented in Scrum. SAFe®, however, introduces a different structure and new conceptions. SAFe® and LeSS both offer a new concept to handle large projects, which require hundreds or thousands of people.

5.3. Limitations

After the detailed description and comparison of SAFe® and LeSS, the following limitations can be crystallized out:

- **Comparison of two scaling agile frameworks:** Because organizations faced their challenges in scaling agile across the entire IT organizations, several scaling agile frameworks emerged to meet this challenge. In total, 20 scaling agile frameworks were identified. The goal of this thesis was to create a comparison template, based on which each scaling agile framework can be compared. This Bachelor's thesis was limited to compare the two most well-known scaling agile frameworks.

- **Description and comparison of SAFe® 4.0:** During the writing process of this thesis, a new version of SAFe® was released — SAFe® 4.5. Since about this time, all the necessary information about SAFe® 4.0 was collected and evaluated. This thesis treated further on SAFe® 4.0 by describing it in detail and comparing it with LeSS. But to keep the reader up-to-date, the modifications from SAFe® 4.0 to SAFe® 4.5 were described as well.

- **No evaluation of comparison criteria:** In this thesis, the criteria did not result from an evaluation. That means we decided ourselves which criteria would be appropriate and then categorized them. But it is important that such an evaluation is carried out, because this ensures that all criteria worth-knowing are available in the comparison. By that, organizations have a better and faster understanding of the frameworks, and can easily compare them.

5.4. Future Work

This thesis has described the identified scaling agile frameworks. Especially, the focus was on the two most popular ones, namely SAFe® and LeSS. In this context, both frameworks were described in detail. Furthermore, they were compared based on a created comparison template.
The comparison template serves as a basis to compare any scaling agile framework. This should help organizations to better decide which practice fits their organization best. As already mentioned in the limitations, this thesis only compared SAFe® 4.0 and LeSS. This leads to the need to compare further scaling agile frameworks in order to offer a greater bandwidth to the organizations. This statement is also supported by [91]. Here, it is pointed out that, despite the popularity of the topic in the industry, large-scale agile transformations received very little research attention, as almost 90% of the selected papers were experience reports.
Furthermore, besides SAFe® and LeSS, the other existing scaling agile frameworks need to be described and understood in detail. Thereby, the comparison of them is easier to carry out, and the branding bias will be eliminated. This means a deeper analysis is necessary to find the common elements in one scaling agile framework that corresponds with the ones existing in other scaling agile frameworks.
In addition, to provide organizations with the possibility of using the comparisons of scaling agile frameworks, a meaningful processing must be considered.
Finally, interviews and case studies should be conducted with organizations using scaling agile frameworks. These should represent the way of an organization to use a scaling agile framework. In this context, it is interesting to know whether an organization uses a scaling agile framework as an inspiration to develop their own framework, or whether an organization uses the framework as a whole, only a part of it, or a modified version.

Appendix

A. Detailed Descriptions

Here come the details that are not supposed to be in the regular text.

Bibliography

[1] Agile Manifesto - www.agilemanifesto.org. accessed: 2017-08-09.

[2] Dr. dobb's journal's 2008 project success survey - www.ambysoft.com/surveys/success2008.html.

[3] Mayer T (2009).Scrum Roles–an abstraction. http://www.dymaxicon.com/2013/the-peoples-scrum/,.

[4] "Scrum of Scrums". https://www.scruminc.com/scrum-of-scrums/. accessed: 2017-08-09.

[5] "Scrum@Scale™ Certification Now Available". https://www.scruminc.com/scrum-at-scale-certification/. accessed: 2017-08-09.

[6] "Spotify (part 1): What It's Like to Work "The Spotify Way"". http://corporate-rebels.com/spotify-1/. accessed: 2017-08-09.

[7] "The Enterprise Transition Framework". http://www.agile42.com/en/agile-transition/etf/. accessed: 2017-08-09.

[8] Laxmi Ahuja and Rahul Priyadarshi. Agile approaches towards global software engineering. In *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2015 4th International Conference on*, pages 1–6. IEEE, 2015.

[9] Mashal Alqudah and Rozilawati Razali. A review of scaling agile methods in large software development. *International Journal on Advanced Science, Engineering and Information Technology*, 6(6):828–837, 2016.

[10] S Ambler and M Lines. Scaling agile software development tactically: Disciplined agile delivery at scale. *Disciplined Agile Consortium White Paper Series*, 2016.

[11] Scott Ambler and Mark Lines. The Disciplined Agile (DA) Framework. http://www.disciplinedagiledelivery.com/. accessed: 2017-08-09.

[12] Scott Ambler and Mark Lines. The Disciplined Agile (DA) Framework. http://www.disciplinedagiledelivery.com/poster-v2-1/. accessed: 2017-08-09.

[13] Scott W Ambler. The agile scaling model (asm): adapting agile methods for complex environments. *Environments*, 2009.

[14] Scott W Ambler and Mark Lines. *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM Press, 2012.

[15] Scott W Ambler and Mark Lines. The disciplined agile process decision framework. In *International Conference on Software Quality*, pages 3–14. Springer, 2016.

[16] David J. Anderson. *Kanban: evolutionäres Change Management für IT-Organisationen*. dpunkt-Verl., 2011.

[17] Sören Auer and Heinrich Herre. Rapidowlâan agile knowledge engineering methodology. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, pages 424–430. Springer, 2006.

[18] P. Beck, M. Gärtner, C. Mathis, S. Roock, and A. Schliep. ScALeD Agile Lean Development - Die Prinzipien. `http://scaledprinciples.org/de`. accessed: 2017-08-09.

[19] M. Beedle. Enterprise Scrum Definition: Business Agility for the 21st Century. `http://static1.1.sqspcdn.com/static/f/608893/27625791/1500228079760/Enterprise+Scrum+Definition.pdf?token=1rCI5hIdpKWF%2FD1C0tCHaDStbOY%3D`. accessed: 2017-08-09.

[20] M. Beedle. Enterprise Scrum Introduction. `https://medium.com/@mikebeedle/enterprise-scrum-introduction-a4987ee690d0`. accessed: 2017-08-09.

[21] Hilary Berger and Paul Beynon-Davies. The utility of rapid application development in large-scale, complex projects. *Information Systems Journal*, 19(6):549–570, 2009.

[22] Colin Bird and Rachel Davies. Scaling Scrum.

[23] Elizabeth Bjarnason, Krzysztof Wnuk, and Björn Regnell. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In *Proceedings of the 1st Workshop on Agile Requirements Engineering*, page 3. ACM, 2011.

[24] Barry Boehm and Richard Turner. Management challenges to implementing agile processes in traditional development organizations. *IEEE software*, 22(5):30–39, 2005.

[25] Richard Brenner and Stefan Wunder. Scaled agile framework: Presentation and real world example. In *Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on*, pages 1–2. IEEE, 2015.

[26] Em Campbell-Pretty. *"Tribal Unity: Getting from Teams to Tribes by Creating a One Team Culture"*. CreateSpace Independent Publishing Platform, 2016.

[27] A. Cockburn. *"Crystal Clear A Human-Powered Methodology for Small Teams"*. Addison-Wesley Professional, 2004.

[28] A. Cockburn. *"Agile Software Development: The Cooperative Game, Second Edition"*. Addison-Wesley Professional, 2006.

[29] Alistair Cockburn. Using both incremental and iterative development. *STSC CrossTalk (USAF Software Technology Support Center)*, 21(5):27–30, 2008.

[30] David Cohen, Mikael Lindvall, and Patricia Costa. Agile software development. *DACS SOAR Report*, 11, 2003.

[31] Agile Business Consortium. The DSDM Agile Project Framework. `https://www.agilebusiness.org/content/introduction-0`. accessed: 2017-08-09.

[32] Stephen Denning. How to make the whole organization "agile". *Strategy & Leadership*, 44(4):10–17, 2016.

[33] Tom Devos. Case study: Agility at scale wolters kluwer belgium. Master's thesis, UHasselt, 2014.

[34] Kim Dikert, Maria Paasivaara, and Casper Lassenius. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, 119:87–108, 2016.

[35] Torgeir Dingsøyr and Nils Brede Moe. Towards principles of large-scale agile development. In *International Conference on Agile Software Development*, pages 1–8. Springer, 2014.

[36] Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, and Nils Brede Moe. A decade of agile methodologies: Towards explaining agile software development, 2012.

[37] T Dot and AW George. Using an agile approach in a large, traditional project. In *Proceedings of the Conference on AGILE*, 2006.

[38] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9):833–859, 2008.

[39] Tore Dyba and Torgeir Dingsoyr. What do we know about agile software development? *IEEE software*, 26(5):6–9, 2009.

[40] Nina Dzamashvili Fogelström, Tony Gorschek, Mikael Svahnberg, and Peo Olsson. The impact of agile principles on market-driven software product development. *Journal of Software: Evolution and Process*, 22(1):53–80, 2010.

[41] Ulrik Eklund, Helena Holmström Olsson, and Niels Jørgen Strøm. Industrial challenges of scaling agile in mass-produced embedded systems. In *International Conference on Agile Software Development*, pages 30–42. Springer, 2014.

[42] Malte Foegen and Christian Kaczmarek. *Organisation in einer Digitalen Zeit: Ein Buch für die Gestaltung von reaktionsfähigen und schlanken Organisationen mit Hilfe von skalierten Agile Lean Mustern*. wibas GmbH, 2016.

[43] Amani Mahdi Mohammed Hamed and Hisham Abushama. Popular agile approaches in software development: Review and analysis. In *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*, pages 160–166. IEEE, 2013.

[44] William Hayes, Mary Ann Lapham, Suzanne Garcia-Miller, Eileen Wrubel, and Peter Capell. Scaling agile methods for department of defense programs. 2016.

[45] Scaled Agile Inc. SAFe® 4.0 Introduction: Overview of the Scaled Agile Framework® for Lean Software and Systems Engineering. , Scaled Agile Inc., 2016.

[46] VersionOne Inc. 11th Annual State of Agile™Report. , 2017. accessed: 2017-08-12.

[47] Mehul Kapadia. Introduction to Enterprise Agile Frameworks, Salt Lake City, Utah 2014.

[48] Mehul Kapadia. SCALING SCRUM STEP BY STEP: "THE MEGA FRAMEWORK", São Paulo, Brazil 2012.

[49] B Kent and A Cynthia. *"Extreme Programming Explained: Embrace Change, Second Edition"*. Addison-Wesley Professional, 2005.

[50] Richard Knaster and Dean Leffingwell. *SAFe 4.0 Distilled: Applying the Scaled Agile Framework for Lean Software and Systems Engineering*. Addison-Wesley Professional, 2017.

[51] Henrik Kniberg and Anders Ivarsson. Scaling agile@ spotify. *online], UCVOF, ucvox. files. wordpress. com/2012/11/113617905-scaling-Agile-spotify-11. pdf*, 2012.

[52] Janne J Korhonen and Konsta Luhtala. Upping the ante: Agile product development at ray.

[53] Eetu Kupiainen, Mika V Mäntylä, and Juha Itkonen. Using metrics in agile and lean software development–a systematic literature review of industrial studies. *Information and Software Technology*, 62:143–163, 2015.

[54] M Laanti. *Agile Methods in Large-Scale Software Development Organizations*. PhD thesis, Doctoral thesis. Oulu 2012. University of Oulu, Faculty of Science, Department of Information Processing Science, 2012.

[55] Corey Ladas. *Scrumban-essays on kanban systems for lean software development*. Lulu. com, 2009.

[56] Craig Larman. Scrum vs. Kanban vs. Scrumban: Team Members, Meetings, and Roles. `https://less.works/profiles/craig-larman`. accessed: 2017-08-12.

[57] Craig Larman. *Scaling lean & agile development: thinking and organizational tools for large-scale Scrum*. Pearson Education India, 2008.

[58] Craig Larman and Bas Vodde. Architecture & Design. `https://less.works/less/technical-excellence/architecture-design.html`. accessed: 2017-08-12.

[59] Craig Larman and Bas Vodde. Lean primer. *version*, 1:1–46, 2009.

[60] Craig Larman and Bas Vodde. *Practices for scaling lean & Agile development: large, multisite, and offshore product development with large-scale scrum*. Pearson Education, 2010.

[61] Craig Larman and Bas Vodde. Scaling agile development. *CrossTalk*, 9:8–12, 2013.

[62] Craig Larman and Bas Vodde. *Large-scale scrum: More with LeSS*. Addison-Wesley Professional, 2016.

[63] Craig Larman and Bas Vodde. *"Large-Scale Scrum, 1st Edition"*. dpunkt, 2017.

[64] D. Lasaite. Scrum vs. Kanban vs. Scrumban: Team Members, Meetings, and Roles. `http://www.eylean.com/blog/2013/05/scrum-vs-kanban-vs-scrumban-team-members-meetings-and-roles/`. accessed: 2017-08-09.

[65] MM Leeuwen. Agile scaling@ topicus: s caling scrum with help of agile scaling frameworks at topicus finance. Master's thesis, University of Twente, 2015.

[66] D. Leffingwell. Implementing SAFe. `https://www.scaledagile.com/certification/courses/implementing-safe/`. accessed: 2017-08-09.

[67] D. Leffingwell. Leading SAFe. `https://www.scaledagile.com/certification/courses/leading-safe/`. accessed: 2017-08-09.

[68] D. Leffingwell. SAFe Advanced Scrum Master. `https://www.scaledagile.com/certification/courses/safe-advanced-scrum-master/`. accessed: 2017-08-09.

[69] D. Leffingwell. SAFe Blog. `http://www.scaledagileframework.com/blog/`. accessed: 2017-08-09.

[70] D. Leffingwell. SAFe Community. `https://community.scaledagile.com/CustomCommunityLogin`. accessed: 2017-08-09.

[71] D. Leffingwell. SAFe for Teams. `https://www.scaledagile.com/certification/courses/safe-for-teams/`. accessed: 2017-08-09.

[72] D. Leffingwell. SAFe Product Owner/Product Manager. `https://www.scaledagile.com/certification/courses/safe-product-manager-product-owner/`. accessed: 2017-08-09.

[73] D. Leffingwell. SAFe Release Train Engineer. `https://www.scaledagile.com/certification/courses/safe-release-train-engineer/`. accessed: 2017-08-09.

[74] D. Leffingwell. SAFe Scrum Master. `https://www.scaledagile.com/certification/courses/safe-scrum-master/`. accessed: 2017-08-09.

[75] Dean Leffingwell. Enterprise Scrum Introduction. `http://www.scaledagileframework.com/whats-new-in-safe-45/`. accessed: 2017-08-09.

[76] Dean Leffingwell. Safe ® 4.0 for Lean Enterprises. http://www.scaledagileframework.com/. accessed: 2017-08-09.

[77] Dean Leffingwell. Safe ® 4.0 for lean software and systems engineering. http://v4.scaledagileframework.com/. accessed: 2017-08-09.

[78] Dean Leffingwell. *Scaling software agility: best practices for large enterprises*. Pearson Education, 2007.

[79] Dean Leffingwell. *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.

[80] Dean Leffingwell. *SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering*. Addison-Wesley Professional, 2016.

[81] Dean Leffingwell. SAFe® 4.0 Introduction, subtitle=Overview of the Scaled Agile Framework® for Lean Software and Systems Engineering. Technical report, Scaled Agile, 2016.

[82] Mikael Lindvall, Dirk Muthig, Aldo Dagnino, Christina Wallin, Michael Stupperich, David Kiefer, John May, and Tuomo Kahkonen. Agile software development in large organizations. *Computer*, 37(12):26–34, 2004.

[83] Rafael P Maranzato, Marden Neubert, and Paula Herculano. Scaling scrum step by step:" the mega framework". In *Agile Conference (AGILE), 2012*, pages 79–85. IEEE, 2012.

[84] E. Marks. "Governing enterprise agile development without slowing it down: Achieving friction-free scaled agile governance via event- driven governance," AgilePath Corporation, Tech. Rep., 2014.

[85] Poppendieck Mary and Poppendieck Tom. Lean software development: an agile toolkit, 2003.

[86] Peter Merel. The Manifesto for Agile Organization. http://xscalealliance.org/. accessed: 2017-08-09.

[87] Peter Merel. XSCALE Structures. http://xscale.wiki/#XSCALE%20Structures. accessed: 2017-08-09.

[88] John Noll, Abdur Razzak, Ita Richardson, and Sarah Beecham. Agile practices for the global teaming model. In *Global Software Engineering Workshops (ICGSEW), 2016 IEEE 11th International Conference on*, pages 13–18. IEEE, 2016.

[89] Uludağ Ö., Kleehaus M., X.Xu, and Matthes F. : Investigating the Role of Architects in Scaled Agile Frameworks, 21th Conference on Enterprise Distributed Object Computing (EDOC), Québec City, Canada 2017.

[90] Maria Paasivaara, Sandra Durasiewicz, and Casper Lassenius. Using scrum in a globally distributed project: a case study. *Software Process: Improvement and Practice*, 13(6):527–544, 2008.

[91] Maria Paasivaara and Casper Lassenius. Challenges and success factors for large-scale agile transformations: A research proposal and a pilot study. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, page 9. ACM, 2016.

[92] Maria Paasivaara and Casper Lassenius. Scaling scrum in a large globally distributed organization: A case study. In *Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference on*, pages 74–83. IEEE, 2016.

[93] Frauke Paetsch, Armin Eberlein, and Frank Maurer. Requirements engineering and agile software development. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*, pages 308–313. IEEE, 2003.

[94] S. Pahuja. "What is Scrumban?". `https://www.agilealliance.org/what-is-scrumban/`. accessed: 2017-08-09.

[95] Akhila Pant. Agile methodology. 2016.

[96] Georgios Papadopoulos. Moving from traditional to agile software development methodologies also on large, distributed projects. *Procedia-Social and Behavioral Sciences*, 175:455–463, 2015.

[97] Kai Petersen. Is lean agile and agile lean. *Modern Software Engineering Concepts and Practices: Advanced Approaches*, page 19, 2010.

[98] Kai Petersen and Claes Wohlin. The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering*, 15(6):654–693, 2010.

[99] Ron Quartel. About FAST Agile. accessed: 2017-08-09.

[100] Ron Quartel. Roles. `http://www.fast-agile.com/method/fast-roles`. accessed: 2017-08-09.

[101] Asif Qumer and Brian Henderson-Sellers. A framework to support the evaluation, adoption and improvement of agile methods in practice. *Journal of Systems and Software*, 81(11):1899–1919, 2008.

[102] Dolman R. and Spearman S. Agile scaling knowledgebase decision matrix. `http://www.agilescaling.org/ask-matrix.html`. accessed: 2017-08-09.

[103] D. Radigan, B. Huff, and S. Jain. Scaling agile with Atlassian and SAFe®. Technical report, Atlassian, 2016.

[104] Mohammad Abdur Razzak. Transition from plan-driven to agile: An action research. In *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17*, pages 746–750. Springer, 2016.

[105] Ajay Reddy. *The Scrumban [r] evolution: Getting the Most Out of Agile, Scrum, and Lean Kanban*. Addison-Wesley Professional, 2015.

[106] Darrell K Rigby, Jeff Sutherland, and Hirotaka Takeuchi. Embracing agile. *Harvard Business Review*, 94(5):40–50, 2016.

[107] Hina Saeeda, Fahim Arif, Nasir Mehmood Minhas, and Mammona Humayun. Agile scalability for large scale projects: Lessons learned. *JSW*, 10(7):893–903, 2015.

[108] Ken Schwaber. Nexus™Guide: *The Definitive Guide to Nexus: The exoskeleton of scaled Scrum development*. https://www.scrum.org/resources/online-nexus-guide. accessed: 2017-08-09.

[109] Ken Schwaber. *The enterprise and scrum*. Microsoft Press, 2007.

[110] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[111] Ken Schwaber and Jeff Sutherland. The scrum guide. *Scrum Alliance*, 2011.

[112] Scrum.org. The Scrum Framework Poster. https://www.scrum.org/resources/scrum-framework-poster?gclid=EAIaIQobChMIqcLLspbb1QIVor_tCh2aEAHwEAAYASAAEgKQNfD_BwE. accessed: 2017-08-09.

[113] Fernando Selleri Silva, Felipe Santana Furtado Soares, Angela Lima Peres, Ivanildo Monteiro de Azevedo, Ana Paula LF Vasconcelos, Fernando Kenji Kamei, and Silvio Romero de Lemos Meira. Using cmmi together with agile software development: A systematic review. *Information and Software Technology*, 58:20–43, 2015.

[114] Andy Singleton. How SAFe agile kills innovation. https://blog.maxos.ai/how-safe-agile-kills-innovation-514673f6855e. accessed: 2017-08-09.

[115] Andy Singleton. Matrix of Services. http://www.continuousagile.com/unblock/ea_matrix.html. accessed: 2017-08-09.

[116] Andy Singleton. Meet the Apex Competitors of Capitalism. https://blog.maxos.ai/meet-the-apex-competitors-of-capitalism-75051baa9c21. accessed: 2017-08-09.

[117] Andy Singleton. Unblock! A Guide to the New Continuous Agile. http://www.continuousagile.com/unblock/ea_matrix.html. accessed: 2017-08-09.

[118] Igor Stojanov. Scaling agile using scaled agile framework. 2015.

[119] Igor Stojanov, Oktay Turetken, and Jos JM Trienekens. A maturity model for scaling agile development. In *Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on*, pages 446–453. IEEE, 2015.

[120] Jeff Sutherland and Alex Brown. Scrum at Scale Framework™. https://www.scruminc.com/scrum-scale-case-modularity/. accessed: 2017-08-09.

[121] K Thompson. Recipes for agile governance in the enterprise, 2013.

[122] Kevin Thompson. RECIPES FOR AGILE GOVERNANCE IN THE ENTERPRISE. https://www.cprime.com/rage/. accessed: 2017-08-09.

[123] Oktay Turetken, Igor Stojanov, and Jos JM Trienekens. Assessing the adoption level of scaled agile development: a maturity model for scaled agile framework. *Journal of Software: Evolution and Process*, 29(6), 2017.

[124] Aashish Vaidya. Does dad know best, is it better to do less or just be safe? adapting scaling agile practices into the enterprise. *PNSQC. ORG*, pages 1–18, 2014.

[125] Arie van Bennekum. The DSDM process. https://www.agilebusiness.org/content/process. accessed: 2017-08-12.

[126] Satisha Venkataramaiah. Lean Enterprise Agile Framework - A Systems Thinking approach to scale deliverables. https://confengine.com/agile-india-2016/proposal/1677/. accessed: 2017-08-09.

[127] Satisha Venkataramaiah. Lean Enterprise Agile Framework LEAF. https://leanpitch.com/lean-enterprise-agile-framework-leaf/. accessed: 2017-08-09.

[128] Kati Vilkki. When agile is not enough. In *Lean Enterprise Software and Systems*, pages 44–47. Springer, 2010.

[129] B. Vodde and C. Larman. Communities. https://less.works/resources/communities.html. accessed: 2017-08-09.

[130] B. Vodde and C. Larman. LeSS– More with LeSS. https://less.works/less/framework/index.html. accessed: 2017-08-09.

[131] B. Vodde and C. Larman. LeSS– More with LeSS. https://less.works/. accessed: 2017-08-09.

[132] B. Vodde and C. Larman. LeSS Courses. https://less.works/courses/less-courses.html. accessed: 2017-08-09.

[133] B. Vodde and C. Larman. The LeSS Blog. https://less.works/blog/index.html. accessed: 2017-08-09.

[134] Jan Vom Brocke, Alexander Simons, Bjoern Niehaves, Kai Riemer, Ralf Plattfaut, Anne Cleven, et al. Reconstructing the giant: On the importance of rigour in documenting the literature search process. In *ECIS*, volume 9, pages 2206–2217, 2009.

[135] D. Wells. Extreme Programming: A gentle introduction . http://www.extremeprogramming.org/. accessed: 2017-08-09.

[136] Mike West, Nathan Wilson, and Stefan Van Der Zijden. Market Guide for Enterprise Agile Frameworks, 2016.

[137] Laurie Williams and Alistair Cockburn. Guest editors' introduction: Agile software development: It's about feedback and change. *Computer*, 36(6):39–43, 2003.

[138] Alex Yakyma. *"The Rollout: A Novel about Leadership and Building a Lean-Agile Enterprise with SAFe ®"*. Addison-Wesley Professional, 2016.