Haaga-Helia
University of Applied Sciences

Scaling Agile Methods

Petteri Karesma

Bachelor's Thesis
Degree Programme in ICT
2016

Haaga-Helia
University of Applied Sciences

Abstract

21 November 2016

Author
Petteri Karesma

Degree programme
Degree Programme in ICT

Thesis title
Scaling Agile Methods

Number of pages and appendix pages
22 + 2

The purpose of this thesis was to give an impartial view on some of the prevailing scaling agile frameworks. As well as provide a general portrayal of how agile scaling is generally done and what it means.

The thesis was compiled by reviewing material found online from various different individuals and resources. The thesis goes through the basic principle of scaling agile as well as few of the most prevailing frameworks supporting it. Chosen frameworks are also studied by reviewing case studies concerning them.

The report showed that implementing a framework to scale agile might be challenging and often proper training and coaching is needed. The study also showed that frameworks have differences that need to be considered when choosing the one to implement.

The report concludes that scaling agile already has and will continue to have an important role in software and product development. Some frameworks are better suited in some environments than others and some environments fit in multiple different frameworks.

Keywords
Scaling, Agile, LeSS, SAFe, Spotify, Scrum

Haaga-Helia
University of Applied Sciences

Abstract

21 November 2016

Tekijä
Petteri Karesma

Koulutusohjelma
Tietojenkäsittely

Opinnäytetyön otsikko
Skaalautuvat ketterät menetelmät
(Scaling Agile Methods)

Sivu- ja liitesivu-määrä
22 + 2

Tämän työn tarkoitus oli antaa puolueeton katsaus huomiota saaneisiin skaalautuviin ketteriin menetelmiin. Sekä tarjota yleiskuvaus siitä kuinka ketterän menetelmän skaalaus yleensä tehdään ja mitä se ylipäätään tarkoittaa.

Työ on koostettu käymällä läpi usean eri henkilön ja resurssin julkaisemaa internetistä löytyvää materiaalia. Tutkimus selittää skaalautuvan ketterän mentelmän käsitteen sekä muutaman sitä tukevan viitekehyksen toimintaperiaatteen. Valittuja viitekehyksiä tutkittiin myös niihin liittyvien käytännön esimerkkitapausten kautta.

Selvitys osoitti että skaalautuvan ketterän menetelmän käyttöönotto ja toteuttaminen voi olla haastavaa ja monesti oikeanlainen kouluttaminen ja valmentaminen on tarpeellista. Työ osoitti myös sen että viitekehyksillä on eroja, jotka pitää ottaa huomioon menetelmää valittaessa.

Selvitys päättelee että skaalautuvat menetelmät ovat ja tulevat olemaan tärkeässä roolissa ohjelmisto- ja tuotekehityksessä. Jotkin viitekehykset soveltuvat paremmin joihinkin ympäristöihin kuin toiset ja jotkin ympäristöt sopivat moneen eri viitekehykseen.

Asiasanat
Scaling, Agile, LeSS, SAFe, Spotify, Scrum

Table of contents

1 Introduction

In this thesis I explore the phenomenon of scaling agile frameworks and delivery methods. I chose to focus on three different agile scaling methods for this study and see how they work and approach the subject. There are multiple different frameworks available but I chose these three because I felt they are the most prominent ones at the moment and provoking most of the discussion.

In the first section of this study I will provide an overview on scaling agile as a whole and on the chosen frameworks individually in hopes to make it easier to understand the concept behind it. But in this thesis I expect that the basic procedures and principles of agile development, like scrum, are already understood and assimilated by the reader.

There are actually quite many case studies, at least from the better known frameworks like Large-Scale Scrum (LeSS) and Scaled Agile Framework (SAFe). In the second part I will go through one case study per each framework I cover in this thesis, and that way try to give a clearer picture of how the said framework works and how it's implemented.

In the last part of the study I will analyse the findings and form a conclusive summary of it. I personally have no experience in working with scaled agile models but I believe that will help me to form an unbiased opinion of the methods. This thesis is meant to present an introductory portrayal of scaling agile to those that want to learn what it means. And perhaps generate some thoughts on what to keep in mind when choosing which method to implement.

2 Scaling Agile

Research foresees that agile methodologies will be needed in the majority of the software projects by 2018. So agile is not used just by startups anymore, but bigger enterprises too. And to solve the challenges that come when implementing agile in large organizations, it has to scale. (Weston Rowell 2016.)

Usually five to ten persons makes a typical team in agile environment, and it's the same in scaled agile. Instead of adding more people in teams you increase the number of teams. That's important because it is still the team's responsibility to deliver the results, that fulfill customer's expectations and needs. And also that way teams will stay as self-managing and self-organizing empowered units. The increasing number of teams will need increased transparency, flow efficiency and learning, and for that purpose some layered-in practices are needed. (Weston Rowell 2016.)

Weston Rowell (2016) speaks about three key practices that are:

- Cadence and synchronization.
- Managing WIP (work in process).
- Collaboration in solving the biggest problems.

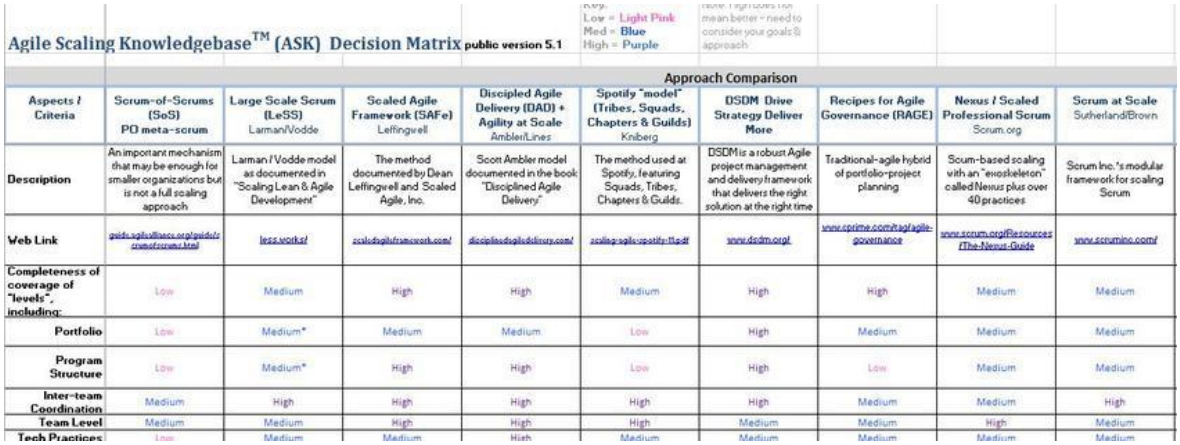
For the teams to have regular chances to adjust, check and plan together, learning cycles and a consistent cadence of planning needs to be established. Synchronization between related programs and teams is also needed. (Weston Rowell 2016.)

Managing WIP becomes even more important when work in progress increases due to more teams being involved. And without good WIP management, flow efficiency can't be maximized. To accomplish this the responsibility for flow management and work prioritization need to be properly assigned to specific roles. Scrum masters and coaches are also in key roles to help teams and organizations reduce friction between all sides. (Weston Rowell 2016.)

Scaling the agile can hamstring the progress and get stressful for teams when facing dependencies and trying to understand their own role in the bigger picture. For that it is extremely important to increase collaboration for the project across the whole organization. One other thing to remember is that different cultures and time zones can be a big challenge in communication and collaboration, so the tools and technology as well as facilitation need to be handled exceptionally well. (Weston Rowell 2016.)

3 Frameworks that Scale Agile

Agile coaches Richard Dolman and Steve Spearman (2014a) compare different scaling agile frameworks in their website. Their goal is to provide comprehensive information about comparing the most prevalent frameworks of scaled agile. They present this information in a comparison matrix and offer it to free use – with proper citation. (Dolman & Spearman 2014a.) Sample image of the comparison matrix is shown below. The whole image can be found in the attachments. And download link for the excel -file of the matrix is found at Dolman and Spearman's website.



Agile Scaling Knowledgebase™ (ASK) Decision Matrix public version 5.1

Low = Light Pink
Med = Blue
High = Purple

mean better = need to consider your goals & approach

	Approach Comparison								
Aspects / Criteria	**Scrum-of-Scrums (SoS)** PO meta-scrum	**Large Scale Scrum (LeSS)** Larman/Vodde	**Scaled Agile Framework (SAFe)** Leffingwell	**Disciplined Agile Delivery (DAD) + Agility at Scale** Ambler/Lines	**Spotify "model" (Tribes, Squads, Chapters & Guilds)** Kniberg	**DSDM Drive Strategy Deliver More**	**Recipes for Agile Governance (RAGE)**	**Nexus / Scaled Professional Scrum** Scrum.org	**Scrum at Scale** Sutherland/Brown
Description	An important mechanism that may be enough for smaller organizations but is not a full scaling approach	Larman / Vodde model as documented in "Scaling Lean & Agile Development"	The method documented by Dean Leffingwell and Scaled Agile, Inc.	Scott Ambler model documented in the book "Disciplined Agile Delivery"	The method used at Spotify, featuring Squads, Tribes, Chapters & Guilds.	DSDM is a robust Agile project management and delivery framework that delivers the right solution at the right time	Traditional-agile hybrid of portfolio-project planning	Scum-based scaling with an "exoskeleton" called Nexus plus over 40 practices	Scrum Inc.'s modular framework for scaling Scrum
Web Link	guide.agilealliance.org/guide/roy/overview.html	less.works/	scaledagileframework.com/	disciplinedagiledelivery.com/	scaling-agile-spotify-11a.pdf	www.dsdm.org/	www.cprime.com/tag/agile-governance	www.scrum.org/Resources/The-Nexus-Guide	www.scruminc.com/
Completeness of coverage of "levels", including:	Low	Medium	High	High	Medium	High	High	Medium	Medium
Portfolio	Low	Medium*	Medium	Medium	Low	High	Medium	Medium	Medium
Program Structure	Low	Medium*	High	High	Low	High	Low	Medium	Medium
Inter-team Coordination	Medium	High	High	High	High	High	Medium	Medium	High
Team Level	Medium	Medium	High	High	High	Medium	Medium	High	Medium
Tech Practices	Low	Medium	Medium	High	Medium	Medium	Medium	Medium	Medium

Image 1. ASK: Agile Scaling Knowledge – The Matrix (Dolman & Spearman 2014b)

I will cover few of these frameworks that Dolman and Spearman compares in their matrix and compare my findings with their criteria in mind. Dolman and Spearman (2014a) also include email commentary from the authors of the frameworks in their website, and I will reference one of those. Although it should be noted that there is no mention of the year when those inquiries were made. Also noteworthy is that even though Dolman and Spearman talk about 'authors of the frameworks', Henrik Kniberg posted a blog post (2015) about how he is not the inventor of Spotify Model, but how it is the result of everyone trying out different things and collaborating in the Spotify organization.

3.1 Scaled Agile Framework

Scaled Agile Framework, also known as SAFe, was first released in 2011. Since then it has had several major updates and the current up-to-date version is 4.0. SAFe is a knowledge base of patterns for implementing Lean-Agile development at large scale. The framework is freely revealed online. (Scaled Agile, Inc. 2016a.)

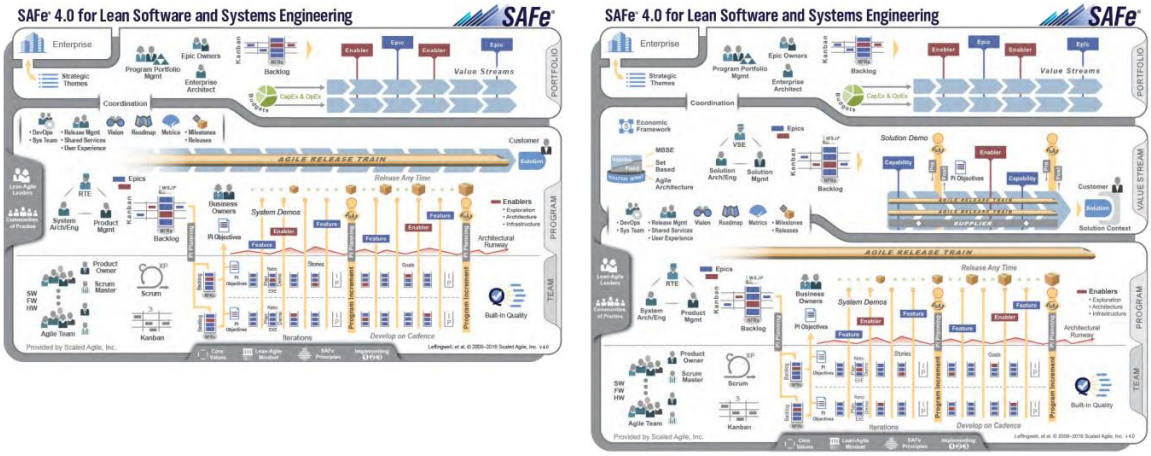


Image 2. 3-level and 4-level SAFe (Scaled Agile, Inc. 2016b, 1)

SAFe can be used with two different types of configurations. The image above shows the overview of those two types. The left side of the image shows the 3-level view, which is good for largely independent smaller systems, products and services, but also supports solutions requiring a modest number of teams. Building and maintaining large, integrated systems engaging hundreds of people is supported by the 4-level view on the right. Interactive version of the picture is provided in the SAFe website, and it allows the user to find information on every topic by clicking the desired icon. (Scaled Agile, Inc. 2016b, 1.)

In addition to the four organizational levels in SAFe – shown in the picture above and explained below – SAFe also includes a Foundation layer (Scaled Agile, Inc. 2016b, 2).

- **Team level** – SAFe teams are agile teams that use Scrum or Kanban methods in iterative development environment. Practices are derived from eXtreme Programming and Lean product development.
- **Program level** – 5 to 12 teams are put together to make a team of teams called an Agile Release Train (ART). This virtual program structure provides alignment, guidance and facilitation.
- **Value Stream level** – Large and complex solution development that require multiple synchronized ARTs is supported by Value Stream level.
- **Portfolio level** – Provides funding to solution development and also funds and organizes value streams.
- **Foundation layer** – Includes elements like Lean-Agile Leaders, Core Values and Principles among others that support development.

SAFe has four core values, which are alignment, built-in quality, transparency and program execution. These values work as a guideline on how to behave and take action as well as help people target their focus and do the right things. Values also help the company stay on the right path. (Scaled Agile, Inc. 2016b, 3.)

"SAFe's practices are grounded on nine fundamental principles that have evolved from Agile principles and methods, Lean product development, systems thinking, and observation of successful enterprises." (Scaled Agile, Inc. 2016b, 6). These Lean-Agile Principles are listed below verbatim (Scaled Agile, Inc. 2016b, 6-8).

- 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 of knowledge workers
- Decentralize decision-making

3.2 Large-Scale Scrum

In 2005 increasing requests and their own interest to apply Scrum to offshore, multi-site and large development projects, led Craig Larman and Bas Vodde teaming up and work with clients to scale up Scrum. And nowadays big groups in different domains worldwide have adopted the two Large-Scale Scrum (LeSS) frameworks – 'smaller' LeSS and LeSS Huge. (The LeSS Company B.V. 2016a.)

3.2.1 LeSS Framework

“Scaled Scrum is not a special scaling framework that happens to include Scrum only at the team level. Truly scaled Scrum is Scrum *scaled*” (The LeSS Company B.V. 2016a). The idea of LeSS is to find out how, and then incorporate the purpose, principles, elegance and elements of Scrum as simply as possible in a large scale development. (The LeSS Company B.V. 2016a.)

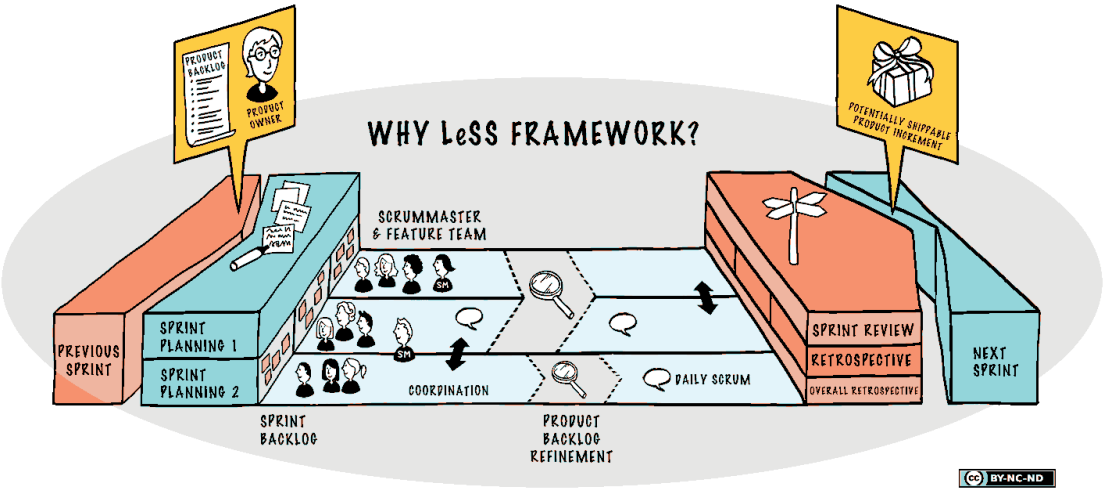


Image 3. LeSS Framework (The LeSS Company B.V. 2016b)

The image above shows the basic structure of LeSS. Because LeSS is one-team Scrum scaled it uses lot of the same ideas and practices as basic Scrum. For example, it has one Product Owner and a single Product Backlog. For every team there’s one common Definition of Done and at the end of every sprint there’s one shared Potentially Shippable Product Increment. (The LeSS Company B.V. 2016b.)

3.2.2 LeSS Huge

LeSS Huge is meant to be used when development project has more than eight teams. LeSS Huge basically means that LeSS is scaled up even further by stacking multiple smaller LeSS frameworks on top of each other (Image 4). (The LeSS Company B.V. 2016c.)

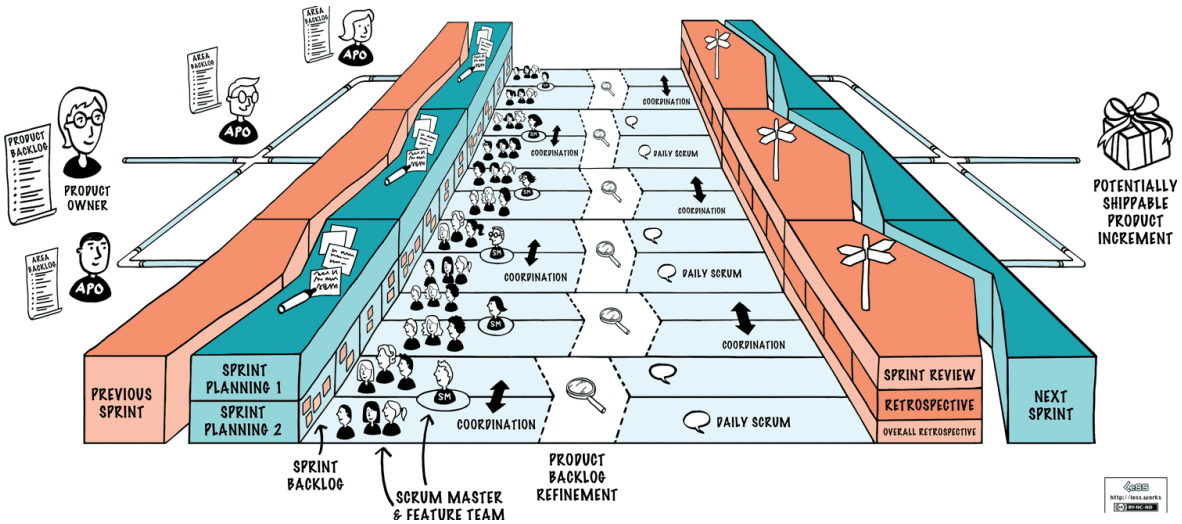


Image 4. LeSS Huge Framework (The LeSS Company B.V. 2016c)

In LeSS Huge there's still only one Potentially Shippable Product Increment but couple changes in roles are needed. Every stacked up LeSS Framework has a product owner called Area PO and above them is a Product Owner who oversees the whole project. There is no single backlog created for the whole project, instead every stacked up LeSS has their Area Backlog. And understandably with more people in the project, organizing and having meetings require some changes. (The LeSS Company B.V. 2016c)

3.3 Spotify "Model"

"Spotify's approach to agile scaling is not really about our process or structure, it's all about the culture" (Kniberg). Spotify's scaling method may not be the easiest one to adopt in other organizations but is still worth discussing about. Although the described scaling method to agile in Spotify was actually never meant to be a framework, just a depiction of their agile journey's present shape. (Kniberg.)

Spotify Model's uniqueness lies in their principle of having very few mandatory practices or hard rules, and their pursuit of aligned autonomy. With aligned autonomy in practice they believe it's possible to keep the autonomy of teams, but still stay collaborative and continue working to achieve the same high-level goal. Spotify Model also aims to have short interval between releases to collect user data more often, so that developers can innovate and improve the product faster. Another important thing is not to utilize failure avoidance but to invest in failure recovery. (Kniberg.)

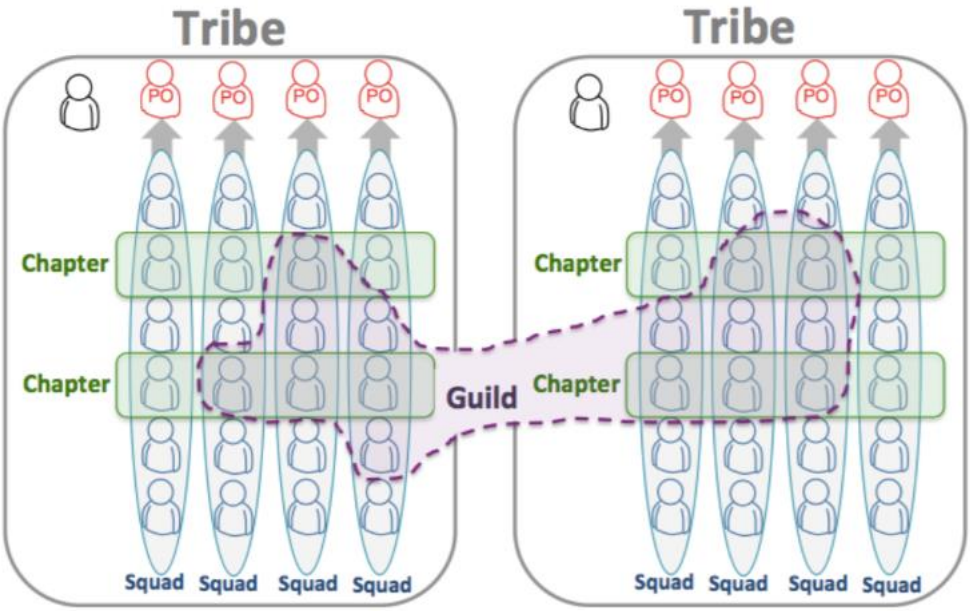


Image 5. Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds (Kniberg & Ivarsson 2012)

Spotify's Scaling Agile method has its own way of addressing and organizing its workforce (Image 5). At Spotify a development team is called a Squad. Squads are self-organizing and get to decide themselves how they want to work. Though they do have their own long-term mission to take responsibility for. As the picture above shows, a Tribe consists of multiple squads. All the squads in the same tribe work in a related field. Chapter is made of people that have similar skillsets and are working in the same tribe in similar capacity. Guild is like a community with people that share same interests. They share things like knowledge, tools and code with each other. (Kniberg & Ivarsson 2012.)

4 Existing Studies on Agile Scaling

In this section I will go through case studies involving the aforementioned frameworks. It will give a more in depth view on how these frameworks function and make it easier to analyse the different features and practices they implement. Worth mentioning is that the referenced material involving the Spotify Model is not really a case study, and is made by Henrik Kniberg, who is one of the many people behind the realisation of Spotify Model. But I deem it beneficial for this thesis because it gives very detailed depiction of how the method works. Besides there's not much case studies available involving Spotify Model.

4.1 SAFe at Nordea

In 2014 Nordea decided to improve their digital banking services and concluded that adopting agile development methodology would be the best way to achieve it. To provide coaching and training during the adoption and development process they chose to use the services of Ivar Jacobson International. So in that year Nordea was introduced to SAFe in a two-day session with management and stakeholders. (Ivar Jacobson International 2015, 1.) Total of 80 people from two existing delivery streams formed Nordea's Agile Release Train that was composed of five development teams, a system team and several other roles (Ivar Jacobson International 2015, 2).

Using simulation-based training and exercises they realized that the proper way to start would be on a single program. The training was also useful for Program Increment planning by helping teams to prepare program backlog better and giving Release Train Engineers (RTEs) support in practical matters concerning the PI planning. (Ivar Jacobson International 2015, 2.) According to the Head of Test & Quality Management Maria Lloyd of Nordea's Digital Banking in the Nordics, all this increased their efficiency by helping everyone understand the work cycle and thereby know what to do and how to do it (Ivar Jacobson International 2015, 3).

4.1.1 Development Process

Development was done in 10 week cycles. One cycle, called Program Increment, consisted of a release planning session, four development iterations, an innovation iteration and a planning iteration. Each increment was planned in a Nordea RTE facilitated PI planning session with the teams and key stakeholders present. There was also a consultant from Ivar Jacobson International (IJI) attending to provide support. (Ivar Jacobson International 2015, 2.) To complete the first PI planning IJI provided the needed knowledge in a training session. All members participated in the first and second PI planning to identify interdependencies, so that they could establish objectives for team-level and program-level increments. (Ivar Jacobson International 2015, 3.)

After that with each PI they continued to evolve and learn. They structured their work in a way that it could be planned without creating premature work breakdown structure or big upfront designs. They also created the backlog on their own instead of getting a list of tasks and features. In PI sessions they identified dependencies with a proper plan and broke down the features themselves. All in all, the delivery system greatly improved. (Ivar Jacobson International 2015, 3.)

The management also became agile, and now on the management level they have a program management team and a program portfolio management board, with both having their own backlog. They also have a program management portfolio on a program level. Even though their backlog may differ from that of a normal agile team's, they still work similarly to the development teams. (Ivar Jacobson International 2015, 4.)

4.1.2 Learning Points

One big benefit of PI sessions was that it helped identify dependencies between teams. And overall implementing agile principles in practice was enabled by PI sessions. At the beginning when the two delivery streams were combined to form the release train, teams were frustrated and didn't work in unison. It got working only after the first PI planning session. They learned that you need people to elaborate the features and transfer knowledge to the teams so that they can pick the required features when needed. (Ivar Jacobson International 2015, 4.)

The program and SAFe implementation was a success and inspired Nordea to scale agile in other departments too. Maria Lloyd believes that they can now confidently continue forward and give best possible support to the agile teams, and deliver value to customers. (Ivar Jacobson International 2015, 4.)

4.2 LeSS at Nokia Siemens Networks

When first discussions about building a high capacity network gateway for Nokia Siemens Networks were happening in 2007, it was decided to choose LeSS as a framework to use. There were two major risks is this project and LeSS seemed to be a suitable way to manage this. The risks were that it was a completely new technology and had never been used in the company before that, and learning based feature content needed to be adapted throughout the process. Choosing a framework capable of agile development proved advantageous in the first few months already. They realized that the broadband network gateway in development needed to be changed for a mobile network that enabled 2G/3G and long-term evolution (LTE), so the flexibility of agile framework made the change of direction go smoothly. (Nyman 2015.)

Even though it was easy to choose LeSS as the framework to use in development, it needed hard work to adopt in practice. Convincing everyone involved that the wanted reduction in cycle time and increase in flexibility would be achieved by preferring feature teams over component teams was the first challenge. Forming the teams had some difficulties at first because the people thought that feature teams lead to a bad software quality. But after some discussion with an agile coach present they agreed to try feature teams and see what happens to the quality of the software. Scrum Masters were chosen from the teams after they realised that the managers chosen before as Scrum Masters were not capable of this role. Instead the managers started to help in other ways by working as free agents. (Nyman 2015.)

After that the development began. They started with two teams, and they had people with traditional waterfall experience and people with scrum experience. But having people with two different backgrounds in the teams created arguments within teams and even more so between teams. They had difficulties to agree on how the architecture and infrastructure should be built because of many dissenting opinions. Even though they managed to help the situation a little by having a shared design session, there still was so many moving parts that the teams actually created conflicting solutions. The positive outcome was that by trying to convince the other team that their solution is better, they actually covered multiple possible ways to design the architecture. Other thing they had difficulties in the beginning was that the teams felt that they couldn’t plan things enough. But the thing was that the development issue they had couldn’t be solved with any amount of planning; the only option was the work, and design and plan when new issues arise. (Nyman 2015.)

4.2.1 Implementing LeSS Framework

In the first LeSS implementation they had a joint planning session held with both teams present for both Sprint Planning 1 and 2. The collaboration between teams helped defining the architecture and solve dependencies. Each team had their own Sprint Backlog and those were managed by visualizing it on a wall. This ‘cards on a wall’ method proved popular and all later teams adopted it too. At first Product Backlog Refinement (PBR) was done by the teams themselves during the sprints. But even though they could do this since requirements and priorities were clear in the beginning, there were conflicts between teams because they lacked collaboration in refinement. Later they changed the practice so that each team would meet the Product Owner and have a proper PBR session. At the end of a sprint they arranged a Sprint Review with both teams involved, individual Retrospectives for each team and a common Overall Retrospective with both teams. (Nyman 2015.)

When they added the total number of teams to four, they faced challenges again because the people were transferred from a traditional organization. For several sprints one team couldn’t produce anything that could be considered as done, because they refused to adopt the new testing tools and the new way of doing things. It was acknowledged that there should have been more training and reasoning provided to the people in the ways of how iterative development works. Around this time, they also added people to do the documentation who worked with the teams directly during sprints. (Nyman 2015.)

Continuing after integrating the new teams, the project-wide Sprint Planning 1 and team-level Sprint Planning 2 were held in the same place with all team members present. This was made to increase collaboration and relieve the decision process in the planning. Sprint Reviews went okay, even with six teams, but the Overall Retrospective was a big challenge. Improvements were happening slowly because teams were inefficient and managers lacked real focus and often immediate control of the improvements. (Nyman 2015.)

4.2.2 Moving to LeSS Huge

At the next phase they started to add teams at a second site. They wanted to speed up the development with more teams because the market demand was emerging for the product. They chose to use a subcontractor to provide the teams, and then trained them by putting them to work as team members in the local teams. Both locations had the same coding and testing rules in practice. The biggest challenge was the communication between the locations. They decided to use a Product Owner proxy to keep the misunderstandings in requirements at a minimum. (Nyman 2015.)

At this phase they saw that one Product Owner became burdened too much, so they started to categorize teams to work in specific requirement areas so that Product Backlog items could be categorized too. It was a step in the right direction, although they did realise later on that they should have put more teams in a particular area, not just two teams for example. The Product Owner was having trouble getting more detailed view on what each team was doing, so they tried to add some Area Product Owners but without success. It failed mainly because the Product Owner didn't want to give anyone else much authority to make decisions. So they spent considerable amount of time trying to convince specification people in product management to work with the teams. They ended up in a situation where teams would work with several feature experts corresponding to their feature area. (Nyman 2015.)

At this point, the Sprint Planning 1 was already working well. Since Product Backlog Refinement was done properly with the feature experts, there only had to be a few representatives from every team present. But by the time there was more than eight teams on one site, it started to feel too heavy to have only one big Sprint Review. So they changed it to a series of smaller and shorter meetings. Product Owner and feature experts would visit every team in turn, and they would present what they had done. And other teams could follow the reviews that they were interested in. Product Owner proxy did the review for the remote teams in a separate event and others followed it using teleconferencing. (Nyman 2015.)

4.2.3 Structure and Process Thereafter

Eventually they got to over 20 teams. Most of those teams are developing and documenting features, but they also have some teams in supporting roles like Continuous Integration System (CIS) team. Some other examples include teams that concentrate on things like coaching and different kinds of testing. Most of the managers that were working as free agents before actually formed a team and started to take work from the Product Backlog, which worked because most of the managers had a strong background in software development. (Nyman 2015.)

Even though a few managers – that didn't want to be in the development team – focused on Scrum Master management, there was still some traditional project managers that didn't really have anything to do. They sometimes helped Product Owner and helped remove impediments, but mostly they just had free time. (Nyman 2015.)

And although the overall LeSS adoption worked, they couldn't apply it to the whole organization. It worked on that project and it changed the lowest level of the organization, which now consists of feature teams, but the first level and above still remained the same with traditional team managers and so on. (Nyman 2015.)

4.3 Spotify Engineering Culture

In 2008 when Spotify was first launched the organization basically just used Scrum. But over time when it grew and more and more teams were added, some of the Scrum practices began hindering the development process. And so changes needed to happen. For them agile and agile principles were more important than Scrum and specific practices, so they decided that rules can be used as a starting point, but they can be broken if needed. So autonomy became their main force to move forward. (Kniberg 2014a.)

Earlier it was mentioned how Spotify consists of autonomous squads. Naturally those squads do have some limits in what they can do, like the overall mission and product strategy and negotiated short term goals. But within those limits they can do pretty much anything they want and decide themselves what to build and how. The autonomy keeps the teams motivated and fast. It also minimizes the bureaucracy so scaling becomes easier. But squads also must be aligned with other squads, product strategies and company's priorities. An individual squad can't be considered more important than Spotify's overall mission, so that's where aligned autonomy comes into place. (Kniberg 2014a.)

4.3.1 Structure and Architecture

Spotify focuses on keeping themselves as a community and try to avoid hierarchical structures. That's the reason for the aforementioned Tribes, Chapters and Guilds. In addition to these the squads are divided in three different types with their own specialty; Client App Squads, Feature Squads and Infrastructure Squads. This came to be when they decided to change the original desktop client's architecture to support decoupled releases. Now their software consists of over a hundred separate systems, and each squad can easily release their product directly. That's possible because each system is coded and deployed independently and thus needs less synchronization with other squads and products. To make this even more easier they have Internal Open-Source Model in practice and everybody can edit every system. This of course needs good communication, collaboration and more importantly a culture of peer code review. (Kniberg 2014a.)

Regardless of the different types of squads none are supposed to serve others by putting code into production for others etc. The idea is to avoid handoffs and instead follow the Self-Service Model by enabling others and give them support to make it easier for them to release themselves. To support this and make it easier to manage and sync the releases they use Release Trains and Feature Toggles. Every application has a release train that departs regularly at predefined times. The idea is to keep releases small and frequent, so it happens routinely and easily. Release train releases all the features and feature toggles are used to hide the unfinished ones. Unfinished features are released and hidden so that it will shed light on integration problems that can then be addressed and fixed. It will also make it easier to identify and find other problems and bugs when testing the application. (Kniberg 2014a.)

4.3.2 Development Environment

Spotify keeps their environment fail friendly where fast recovery from failure means more than avoiding the failure. The idea is that you are bound to make mistakes when building something great so it's better to fail fast and learn from it. The learning part in that is very important so failing is generally followed by a post-mortem. Only when all the learnings from the failure are captured so that the problem can be avoided in the future, can the incident ticket be closed. "Fix the process not just the product." And on top of that all squads talk about what is working well and what needs to be improved in retrospectives held every few weeks. All this relates to Spotify's drive to promote a 'culture of continuous improvement'. (Kniberg 2014b.)

On the other hand, even though failing is not frowned upon, the failing has to happen non-lethally or it might be the last thing they ever make. That's why the 'concept of limited blast radius' is strongly promoted. The decoupled architecture supports this environment; even if a mistake is made it normally only affects a small portion of the system and the rest will stay operational. And because there's no handoffs the squad's end to end responsibility generally means that the problem gets fixed fast. Another thing to limit the impact of something going wrong is that it only affects a small percent of all users, because new features are rolled out step by step and rigorously tracked. So the limited blast radius enables squads to experiment a lot and thereby learn fast, when they don't have to use so much time trying to control and predict all possible risks. (Kniberg 2014b.)

So Spotify's approach to product development is based on principles of lean start-up. But because building the wrong thing is always a bad thing, the risk must be taken into account. Spotify tries to mitigate this risk by informing themselves with research when deciding if new proposed deliverable should be built. They try to answer questions like does users want it, what are the benefits of it, how will it impact the user behaviour etc. If prototypes give good feedback from users and it seems like the product is worth building, they make a Minimum Viable Product (MVP). Then the MVP is tested with a small part of the users and monitored closely. The squad improves and modifies the product until they see what they want from the user data. After that it can be gradually rolled out to all users. This process guarantees that only useful products and features are released because non-successful products are simply not rolled out. (Kniberg 2014b.)

Sometimes there's things like marketing events or partner integrations where making delivery commitments are needed, but overall Spotify focuses more on innovation than predictability and strict schedules. That way squads can operate more freely and concentrate more on delivering value and new ideas. Having new ideas and trying out stuff is encouraged by letting everyone have hack time to experiment, build and think up anything they want. It's a fun way to learn new things and once in a while there's something truly great that comes out of it. They also organize companywide 'Spotify Hack Week' twice a year to endorse this behaviour. All in all, Spotify has a very experiment-friendly culture. (Kniberg 2014b.)

4.3.3 Hardships and Challenges

Spotify's culture is also waste-repellent which means that if something works, keep it, but if something hinders the work get rid of it. They also consider big projects a waste, but that's not something you can totally get rid of. Sometimes the benefits of a big project are possibly more worthwhile than avoiding the risk. In those cases, Spotify uses Progress Visualization, Daily Sync Meetings and Demoing the product every week or so to reduce risk and waste. But that's still an area where they are experimenting a lot and need to improve. (Kniberg 2014b.)

One other thing they face challenges is growth pain. As the company grows it needs to balance between chaos and bureaucracy. Agile mind-set and waste-repellent culture are helping them to find the balance and figure out what's the least amount of bureaucracy to stay out of chaos. The fast growth also creates lots of possible problems when today's seemingly great solution causes a new problem tomorrow. But even though Spotify has a long list of problems and pain-points, it's not that big of an issue because they have people who actually do something about it and problems get fixed quite fast. "Healthy culture heals broken process." (Kniberg 2014b.)

5 Discussion

The first thing that I realised when figuring out these frameworks was that it's worth going through at least couple different ones before deciding which one is the best suited for you. As I started with the Spotify Method and to be honest at first I was really impressed and felt that it would be the best method by far. But as I learned more about the other frameworks, I started to see the whole picture and how there's actually lot in common between all these frameworks. So I think that you actually have to find the differences that matter and that way decide the right one to implement in your project. I think that Henrik Kniberg said it well in the presentation about SAFe at LEGO he and Lars Roost gave in GOTO Conferences 2015. He explained that when people learn little bit about something they are fast to form opinions about it, but that doesn't necessarily mean that those opinions are based on facts or experiences. So you have to be careful when you are in the beginning of a learning curve and eager to gain interest and form opinions, in other words when you are at the top of 'Mount Stupid' as Kniberg (& Roost 2015) named this point in his graph.

5.1 Comparing the Frameworks

After writing this study and exploring the ASK matrix I would say that the biggest difference between Spotify Model and the other two frameworks (LeSS and SAFe) is that Spotify doesn't really have that many specific practices and procedures. It's more about the autonomy and people deciding themselves what to do and how to do it. SAFe and LeSS are little bit more alike, but there is differences between those two also. SAFe is more rigid than LeSS or Spotify Model. But SAFe also has more detailed information and support available, maybe due to it being little bit better known than LeSS and especially Spotify Model. SAFe and LeSS being proper frameworks are offering pretty extensive and comprehensive training and coaching. They also have certification programs. When in the case of Spotify Model, you pretty much have to do it on your own.

In their ASK matrix Dolman and Spearman state that SAFe is typically more expensive than any other framework mentioned, but I would argue that it is extremely hard to compare the possible expenses between the frameworks because of the possible variation in implementation environments and means. Though it is stated that SAFe is more focused in large and enterprise scale projects and the other frameworks tend to be implemented in a little bit smaller scale.

5.2 Implementation Phase

One of the most important thing I gathered from this study is that whichever framework you choose to implement you have to have proper coaching and guidance in the process. In the case of Nordea's SAFe implementation Maria Lloyd mentions in the Ivar Jacobson International's (2015) report that the involvement of IJI in the process was of great help to them. And secondly Ran Nyman (2015) acknowledges in his report that they had instances where they didn't provide sufficient training and knowledge, and those proved to be challenging phases in the process. Coaching is also an important role in the Spotify Model.

Another important thing to consider is the organizational structure and how widely you want the organization to change. Changing the methodologies and development practices in a large organization on a big scale won't happen just like that, it could be very hard, especially if the background is in the traditional waterfall development. And I think this relates to what Kniberg said in his email ('Spotify Author Verbatim Feedback') to Dolman and Spearman how it would be hard to transfer Spotify's approach across organizations. In his blogpost Kevin Goldsmith (2014) goes through some important points what to keep in mind when thinking of applying Spotify's method in other organizations. I do agree with Goldsmith that if you want to implement Spotify Model in whole it requires a certain type of environment, like already accustomed in agile methodologies and willing to give autonomy to teams, but personally I don't see why you couldn't incorporate practices and processes from a couple different frameworks, including Spotify's Model, and in a way make your own framework. But is it worth it, when there's already fully functional frameworks available? On the other hand, frameworks could also be looked at being a big toolbox for product development, just like Kniberg and Roost (2015) described SAFe in their presentation, so aren't you going to modify the process anyway? I guess in the end it comes down to how willing the organization is to change and their commitment to scaling agile, so the implementation process needs to be planned and adjusted accordingly.

5.3 The Conclusion

To conclude I would say that agile scaling already has an important role in software development and it will grow even more in the future. I believe that smaller organizations and especially start-ups understand the importance of agile and for them it might be easier so scale in the future. The bigger challenge lies in large organizations; either they don't realise how important and useful agile could be or they might have difficulties to convince the key personnel to commit to the change.

To summarise the three frameworks covered in this thesis: SAFe is suited for large enterprises as it provides extensive guidance and coaching by being already well known in development circles. LeSS is a good choice for a medium size or a big organization as it will support and cope with the growth of the organization. Though SAFe and LeSS could be as good as any other choice concerning smaller companies, I would say that implementing Spotify Model in a smaller company or especially in start-up as early as possible could prove to be extremely beneficial in the long run.

There are more frameworks to scaling agile and this study was just a scratch in the surface, but hopefully this raised some interest in you as a reader and helps cultivate some ideas and questions for further analysis and studying. Personally for me this was an interesting learning experience and definitely helped me to understand scaling agile better.

References

Dolman, R. & Spearman, S. 2014a. Comparing Agile Scaling Frameworks. URL: http://www.agilescaling.org/home.html. Accessed: 25 October 2016.

Dolman, R. & Spearman, S. 2014b. ASK: Agile Scaling – The Matrix. URL: http://www.agilescaling.org/ask-matrix.html. Accessed: 21 November 2016.

Goldsmith, K. 2014. Thoughts on Emulating Spotify's Matrix Organization in Other Companies. URL: http://blog.kevingoldsmith.com/2014/03/14/thoughts-on-emulating-spotifys-matrix-organization-in-other-companies/. Accessed: 16 November 2016.

Ivar Jacobson International 2015. Nordea – A Uniform Heartbeat with Help from Scaled Agile Framework® and IJI. Ivar Jacobson International SA. URL: http://scaledagileframework.com/nordea-case-study-2/. Accessed: 15 November 2016.

Kniberg, H. Spotify Author Verbatim Feedback. URL: https://app.box.com/s/bqcui8p60fn33bs60e2o. Accessed: 16 November 2016.

Kniberg, H. 2014a. Spotify Engineering Culture – Part 1. URL: https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/. Accessed: 26 October 2016.

Kniberg, H. 2014b. Spotify Engineering Culture – Part 2. URL: https://labs.spotify.com/2014/09/20/spotify-engineering-culture-part-2/. Accessed: 27 October 2016

Kniberg, H. 2015. No, I didn't Invent the Spotify Model. URL: http://blog.crisp.se/2015/06/07/henrikkniberg/no-i-didnt-invent-the-spotify-model. Accessed: 26 October 2016.

Kniberg, H & Ivarsson, A. 2012. Scaling Agile @ Spotify with Tribes, Squads, Chapters & Guilds. URL: https://dl.dropboxusercontent.com/u/1018963/Articles/SpotifyScaling.pdf. Accessed: 25 October 2016.

Kniberg, H. & Roost, L. 2015. Is SAFe Evil? URL: https://www.youtube.com/watch?v=TolNkqyvieE. Accessed: 14 November 2016.

The LeSS Company B.V. 2016a. Introduction to LeSS. URL: http://less.works/less/framework/introduction.html. Accessed: 18 October 2016.

The LeSS Company B.V. 2016b. LeSS Framework. URL: https://less.works/less/framework/index.html. Accessed: 25 October 2016.

The LeSS Company B.V. 2016c. LeSS Huge. URL: https://less.works/less/less-huge/index.html. Accessed: 25 October 2016.

Nyman, R. 2015. Developing a High Capacity Network Gateway with LeSS. URL: https://www.infoq.com/articles/network-gateway-less. Accessed: 15 November 2016.

Scaled Agile, Inc. 2016a. Welcome to Scaled Agile Framework® V4.0! URL: http://scaledagileframework.com/about/. Accessed: 3 November 2016.

Scaled Agile, Inc. 2016b. SAFe® 4.0 Introduction – Overview of the Scaled Agile Framework® for Lean Software and Systems Engineering. Scaled Agile, Inc. Boulder. URL: http://scaledagileframework.com/videos-and-presentations/. Accessed: 7 November 2016.

Weston Rowell, R. 2016. How to Scale Agile for Your Enterprise. URL: https://www.youtube.com/watch?v=gmz4yF-XHUA. Accessed: 19 October 2016.

Appendices

Appendix 1. Agile Scaling Knowledgebase™ (ASK) Decision Matrix – Custom Criteria

Sheet1(Custom Criteria)

Agile Scaling Knowledgebase™ (ASK) Decision Matrix — public version 4.4 (beta template) — Key: Low = Light Pink, Med = Blue, High = Purple — www.agilescaling.org

Custom Criteria Questions / Responses

Do WE have existing, internal capabilities?	*example: No. We do not have a consistent or unified concepts or processes relating to agile and lean practices.*
What Problems are WE trying to solve?	*example: We need to scale additional teams AND deal with the complexity of deploying mulitple products across multiple platforms consistently and efficiently.*
What problems are we observing at the Team level?	*example: Coordination and Synchronization of dependencies and re-usability.*
What problems are we observing at the Management level?	*example: Lack of alignment of Portfolio prioritization and resources.*
Our Definitiion of "Scaling"	*example: Means adding more teams versus expanding into other areas of the organization.*
Do WE understand our Culture and can we define it?	*example: If we feel we have a "Command and Control" versus a "Collaborative" Culture, this may influence our overall criteria and how we evaluate each approach.*
Other...	*Insert additional Rows for YOUR Organizational Criteria...*

Richard Dolman: ID KEY WORDS FROM ORGANIZATIONAL CRITERIA TO ALIGN TO EACH OF THE APPROACHES. USERS OF THIS TOOL MAY ADD ADDITIONAL COLUMNS TO INCLUDE ANY OTHER APPROACHES OR FRAMEWORKS THEY CARE TO INCLUDE.

Approach Comparison

Aspects / Criteria	Scrum-of-Scrums (SoS) PO meta-scrum	Large Scale Scrum (LeSS) Larman/Vodde	Scaled Agile Framework (SAFe) Leffingwell	Discipled Agile Delivery (DAD) + Agility at Scale Ambler/Lines	Spotify "model" (Tribes, Squads, Chapters & Guilds) Kniberg	DSDM Drive Strategy Deliver More	Recipes for Agile Governance (RAGE)	Scrum at Scale Sutherland/Brown	Other...	Notes
Description	An important mechanism that may be enough for smaller organizations but is not a full scaling approach	Larman / Vodde model as documented in "Scaling Lean & Agile Development"	The method documented by Dean Leffingwell and Scaled Agile, Inc.	Scott Ambler model documented in the book "Disciplined Agile Delivery"	The method used at Spotify, featuring Squads, Tribes, Chapters & Guilds.	DSDM is a robust Agile project management and delivery framework that delivers the right solution at the right time	Traditional-agile hybrid of portfolio-project planning	Scrum Inc.'s modular framework for scaling Scrum	*description of other approach to be considered//evaluated...*	
Web Link	guide.agilealliance.org/guide/scrumofscrums.html	less.works/	scaledagileframework.com/	disciplinedagiledelivery.com/	scaling-agile-spotify-11.pdf	www.dsdm.org/	www.cprime.com/tag/agile-governance	www.scruminc.com/	0	

Appendix 2. Agile Scaling Knowledgebase™ (ASK) Decision Matrix – Approach Comparison

Sheet2(Approach Comparison)

Agile Scaling Knowledgebase™ (ASK) Decision Matrix — public version 5.1 — Key: Low = Light Pink, Med = Blue, High = Purple — Note: High does not mean better - need to consider your goals & approach — Latest version can always be found at www.agile

Approach Comparison

Aspects / Criteria	Scrum-of-Scrums (SoS) PO meta-scrum	Large Scale Scrum (LeSS) Larman/Vodde	Scaled Agile Framework (SAFe) Leffingwell	Discipled Agile Delivery (DAD) + Agility at Scale Ambler/Lines	Spotify "model" (Tribes, Squads, Chapters & Guilds) Kniberg	DSDM Drive Strategy Deliver More	Recipes for Agile Governance (RAGE)	Nexus / Scaled Professional Scrum Scrum.org	Scrum at Scale Sutherland/Brown	Other...	Notes	
Description	An important mechanism that may be enough for smaller organizations but is not a full scaling approach	Larman / Vodde model as documented in "Scaling Lean & Agile Development"	The method documented by Dean Leffingwell and Scaled Agile, Inc.	Scott Ambler model documented in the book "Disciplined Agile Delivery"	The method used at Spotify, featuring Squads, Tribes, Chapters & Guilds.	DSDM is a robust Agile project management and delivery framework that delivers the right solution at the right time	Traditional-agile hybrid of portfolio-project planning	Scum-based scaling with an "exoskeleton" called Nexus plus over 40 practices	Scrum Inc.'s modular framework for scaling Scrum	*description of other approach to be considered//evaluated...*		
Web Link	guide.agilealliance.org/guide/scrum	less.works/	scaledagileframework.com/	disciplinedagiledelivery.com/	scaling-agile-spotify-11.pdf	www.dsdm.org/	www.cprime.com/tag/agile-gove	www.scrum.org/Resources/The-I	www.scruminc.com/			
Completeness of coverage of "levels", including:	Low	Medium	High	High	Medium	High	High	Medium	Medium		Note that not all Agilists believe there *should* be levels like this but they occur today in most larger orgs. Low may not = bad!	Low
Portfolio	Low	Medium*	Medium	Medium	Low	High	Medium	Medium	Medium		*-In LeSS, Single PO and one Product Backlog is the portfolio view	Medium
Program Structure	Low	Medium*	High	High	Low	High	Low	Medium	Medium		*-Less is Product focused not traditional Program release focused	High
Inter-team coordination	Medium	High	High	High	High	High	Medium	Medium	High			
Team Level	Medium	Medium	High	High	High	Medium	Medium	High	Medium			
Tech Practices	Low	Medium	Medium	High	Medium	Medium	Medium	Medium	Medium			
Popularity / Adoption (new/growing (low) vs. established/leader (high)	High	Medium	High	Low	Low	Low	Low	Low	Low			
Flexibility / Emergence: Prescriptive (low) vs. emergent (high)?	High	High	Low	Medium	High	Medium	High	Medium	Medium		Note that prescriptive may still include options or allow for customization	
Typical Cost to implement	Low	Low	High	Medium	Low	Medium	Low*				Can vary dramatically - usually can be free via a "roll your own" option	
Availability of Details & Support	Low	Medium	High	Medium	Low	Medium	Medium	Low	Low			
What Team level frameworks are supported? (Scrum, Kanban, XP, etc.)	Scrum	Scrum	Scrum / Kanban / specific XP practices "mandated"	Scrum /Lean	Own method though partly Scrum-like	Own Hybrid Agile Scrum method	Scrum /Lean	Scrum	Scrum			
Emphasizes more Central control or distributed?	Distributed with light coordination	Centralized prioritization and distributed coordination	More Central & top-down on ideas but distributed ownership on "how"	Mixed - depending on chosen parts but can be somewhat central	Distributed with different types of coordination	More centeral control and interactions	Adaptive control that is adjusted as needed	Central Product view and distributed remainder	?			
Scale / Target size (small - med - large)	Small	Med - Large	Large - Enterprise	Med - Large	Med - Large	Med - Large	Small Medium Large	Small but Nexus+ can go over 9	Small - Large		Small: < 100 people or 10 teams Med: >100 < 500 people or 50 teams Large Org: >500 people or 100s of teams *Ranges may be changed by anyone using this tool, based on their relative size preferences.	
Used typically by what Organization Types?	Any that are running Scrum	Has 2 suggested structures for different size organizations	Focused on enterprises	Used in many diverse organizations	Only intended for Spotify - perhaps fits other relatively flat & agile orgs	Mature Complex Organization	Adapts to any size. There is no typical oranization.	New, so adoption is unclear	Adapts to any size. There is no typical oranization.			
Focal point (teams/structure - enterprise/ROI)	team/structure Inter-team dependencies	org descaling, team/structure Agile thinking, PO scale via "areas"	team/structure A customizable but prescriptive framework for most aspects of Agile at scale.	team/structure Larger project stages; Technical process gaps for craftsmanship at scale	team/structure Very agile scaling with little overhead; team affiliations; cross-team concern handling	team/structure Communication Paths colaborative teams	Focus is on Patterns that can be used based on the need of the team program or enterprise	Using Scrum concepts and mindset at scale	Billed as a "Modular" Scaling approach			
Software centric - how often used outside of SW or IT?	Could use anywhere you use Scrum	Focused on Software or SW/HW	Focused on Software or SW/HW	Has been used outside of IT	Spotify only	Has been used outside of IT	Focused on Software	Could use anywhere you use Scrum	Could use anywhere you use Scrum			
Big Positives / Key Differentiatiators	simple, standard Scrum focus on dependencies & resolutions	Good PO scaling, strong principle alignment, Non-prescriptive - gives "suggestions"	The "big picture" and completeness; getting Agile "in the door" at large corporations; actively evolving.	Lots of content; strong in areas such as architecture, design and dev ops; incorporates many good models.	very agile, entrepreneurial, distributed teams, low overhead	Very established following in the UK	Fluid and adaptive	Authored by Ken Schwaber	Lightweight authored by Jeff Sutherland			

Agile Scaling Knowledgebase™ (ASK) Decision Matrix

public version 5.1

Key:
Low = Light Pink
Med = Blue
High = Purple

Note: High does not mean better - need to consider your goals & approach

Latest version can always be found at www.agile

Approach Comparison

Aspects / Criteria	Scrum-of-Scrums (SoS) PO meta-scrum	Large Scale Scrum (LeSS) Larman/Vodde	Scaled Agile Framework (SAFe) Leffingwell	Discipled Agile Delivery (DAD) + Agility at Scale Ambler/Lines	Spotify "model" (Tribes, Squads, Chapters & Guilds) Kniberg	DSDM Drive Strategy Deliver More	Recipes for Agile Governance (RAGE)	Nexus / Scaled Professional Scrum Scrum.org	Scrum at Scale Sutherland/Brown	Other...	N[illegible]
Key Risks / Concerns	limited scaling, limited documentation, not clearly defined Not likely "sufficient" for large scale; some differences in implementation	A more "radically agile" approach that may be a hard sell in larger traditional orgs with many layers and specializations.	Little info on "how", most need certified SPCs to implement properly; Seen as prescriptive; not "agile enough" in its structures; "quick start and leave" issues some places	vague in some areas about the "how"; can come across as a bit disjointed. Not prescriptive in lifecycle.	very limited detail about the "how", Not really a framework; may only fit certain cultures	Heavy process overhead.	New approach that is growing and adapting.	New approach that is growing and adapting. Some of the parts are "secret" unless you go to the class.	New approach that is growing and adapting.		
Training / Resource availability	None known; roll your own	Training and coaching network available	Yes, multi-level training & Certifications	Yes, multi-level Certifications	None known; roll your own	Yes, multi-level Certifications	None known read and imlement yoursef for your need.	Scaled Professional Scrum training & certification is available	Courses are offered		
Deployment Approach (how to get started and make it sustainable)	Self-Organize; roll your own	Covered now on their site https://less.works/less/adoption/getting-started.html	Can roll your own but usually done with certified coaches (SPC's) and training.	Roll your own & pick from a large number of possible practices.	None known; roll your own	Training Coaching by the DSDM group.	roll your own	Likely go to class and probably need SPS coaching.	?		
Notes	Often mis-used and turned into a status meeting	Now offering certifications as practitioner or trainer. Many of its aspects are based on a fairly profound de-scaling of the org and removing of most specialist teams.	Beginning to offer new certifications and have more overlap with Scrum Alliance & Scrum.org	DAD is a hybrid approach which extends Scrum with strategies from Agile Modeling (AM), Extreme Programming (XP), Unified Process (UP), Kanban, Lean Software Development, Outside In Development (OID) and several other methods	Doesn't specifically follow other common frameworks such as Scrum.	Steve Messenger is the POC uk.linkedin.com/pub/steve-messenger/1/a97/72a	Kevin Tompson is the POC www.linkedin.com/in/kevinthompsonphd	Still evolving. There is an approach for > 9 teams (Neus+) but it is only taught at SPS class. Currently, practices are also only taught at SPS class and are not publicly available.	Publicly introduced at Agile 2014 in Orlando		
Other...											
Footnotes											

* - popular in discussion but not widely duplicated as best we can tell
** - but not pure in adherence to Scrum or any one framework

*originally named Scaled Agile Differentiator (SAD)
Concept emerged from Scrum Alliance Scrum Coach Retreat - December 2013
Original workshop members include: Richard Dolman, Steven Spearman, Dave Cornelius, Bob Galen, Bryan Stallings, Perry Reinert, Mike Register, Bill Johnson, and Vijay Kumar

Richard Dolman:
INSERT ADDITIONAL ROWS FOR YOUR ORGANIZATIONAL SPECIFIC CRITERIA...

Examples:
Our Culture favors Command and Control versus Cultivation or Entrepreneurial
Our Domain leans toward compliance and regulation versus experimentation and innovation
Our scaling needs mean adding more teams versus expanding