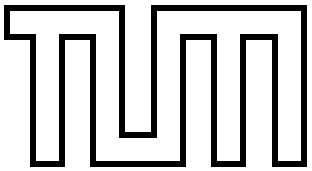


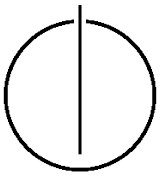
FAKULTÄT FÜR INFORMATIK

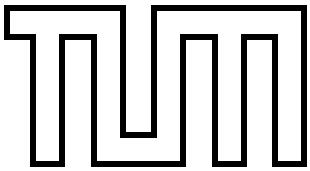
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Wirtschaftsinformatik

The Evolution of Scaling Agile Practices

Gerhard Schwab





FAKULTÄT FÜR INFORMATIK

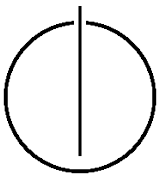
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Wirtschaftsinformatik

The Evolution of Scaling Agile Practices

Die Entwicklung von agil skalierbaren Praktiken

Author:	Gerhard Schwab
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Ömer Uludag
Date:	November 15, 2017



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

München, den 15. November 2017 Gerhard Schwab

Abstract

Since the 1990s the idea of agile problemsolving exists, but only in the recent years with the advent of digitization, the IoT and rapidely changing requirements, have companies recognised its value. To help big companies apply agile methods, that are specifically made for small teams, on a company level, practices for the scaling of those agile methods have been developed.
With this thesis we aim to provide an overview over all Scaling Agile Practices, their evolution and matureness, as well as their foundational methods and practices and the relations between them.
In order to achieve this, we first conducted a literature review identifying 22 different Scaling Agile Practices. Afterwards we invited all main developers of these 22 Scaling Agile Practices to take part in our survey to get the developers view on their work, as well as chronological data and the foundational methods and practices that their Scaling Agile Practices are based on.
This survey was completed by the five Scaling Agile Practices ETF, LeSS, HSD, FAST and the Mega Framework. Their information was then visualized in a Time Map and five Foundational Maps. We found similarities in the challenges they face and their goals and differences in scale and employed approach to acieve the goals, as well as differences in the useage of foundational methods and practices.

Contents

Part I.

Introduction

1. Introduction

In this chapter we give an introduction to this thesis. We convey the motivation behind it, the research questions that follow and the research approach to answer those questions.

1.1. Motivation

With the ongoing digitization of all organizations, the ever ongoing globalization and the increasing speed with wich new enterprise opportunities are presenting themselves, the ability to adjust to change and use it as an advantage has become more and more important. As Eric Overby et al. say: "In turbulent environments, enterprise agility, that is, the ability of firms to sense environmental change and respond readily, is an important determinant of firm success."[55] As a result, agile and lean methods are increasingly applied to the value creation process of organizations, especially in the IT sector. Because of this, there is a high demand for agile practices, as companies largely recognize the value of introducing those methods and practices. [85] As those practices, such as Scrum, Kanban and others, are only intended for small project teams and do not cope well with bigger or multiple teams, big companies with projects involving over one hundred employees have a need for scaled agile practices that can be used at higher levels with bigger numbers of participants, such as LeSS and SaFe. To give a complete overview over those scaled practices and their properties and features, we look at the evolution of scaling agile practices.

1.2. Research Questions

In this section we will describe the research questions (RQ) that this thesis is aiming to answer.

RQ1: Which scaling agile practices exist? In the first research question we give an overview over the current existing scaling agile practices up until the first of august 2017. To achieve this we will conduct an extended literature review (Figure 1.1).The main result answering this research questions will be a list of all existing scaling aigle practices, which will then serve as a basis for all further research questions answered in this thesis.

RQ2: How did scaling agile practices develop? The second research question serves to append chronological data to the information gathered from the first research question. We will gather this data by the extended literature review, as well as an expert interview which we will conduct with the leading developers of scaling agile practices (Figure 1.2). This question will result in the list of existing scaling agile practices being extended with chronological data, as well as a graphic to illustrate the development and matureness of the scaling agile practices.

RQ3: On which foundational methods and practices are scaled agile practices based on? In the last research question we will compare the different existing scaling agile practices with each other. We will complete the overview over all scaling agile practices by showing their foundational methods those practices are based on, as well as show their relationship between each other and the activities,artifacts,and principles those practices employ. To achieve this, we will use the data gathered from the first research question, through the extended literature review, as well as through the expert interview. As a result, we will have a complete overview over the current scaling agile practices based on their foundations and relations between each other. This overview will be visualized in tables as well as in multiple graphics.

1.3. Research Approach

Our approach to answering these three research questions is to first perform a structured literature review and to then conduct an expert interview.
The literature review is conducted as recommended by Brocke et al. [86] ,to ensure the rigour and relevance of this thesis, in a 5 step progress. First we defined the scope of the work and identified the research questions we wanted to answer. After that, we worked out the relevant search terms to find the correct information to answer those questions. We chose the following terms and their synonyms and abbreviations: scaling agile methods, scaling agile practices, scaling agile frameworks, scaling agile software engineering, scaling agile organizations, large-scale agile. Using these search terms we found 612 initial pieces of relevant literatur using the following sources: Science Direct, Web of Science, IEEExplore, ACM Digital library, EBSCOhost, Scopus, SpringerLink, Emerald Insight and GoogleScholar. We conducted the literature search in July/August of 2017. After the initial 612 findings, we analysed the results and synthezised them in multiple revisions. After the first revision, we identified 120 potentially important pieces of literature, which we further reduced to 30 in the second revision. From these 30 papers, books and conference proceedings, the research outcome of the list of the 22 different scaling agile practices was extracted, answering our first research question.
In order to get inside-information, unobtainable by a literature review, we conducted an online expert interview with the developers of the respective scaling agile practices, to answer the second and third research questions. We performed the interview in six phases as recommended by Punter et al. [61]: First we defined the goals of the interview, which we then transformed into a set of questions. Those questions were then transfered over to the unipark tool "questback" [31], with wich we conducted the interview. The link to the questionnaire was then sent out to all 22 scaling agile practice developers. Thereafter we analyzed and interpreted the data, taking into account the results of the literature review. In the last step we visualized the data.

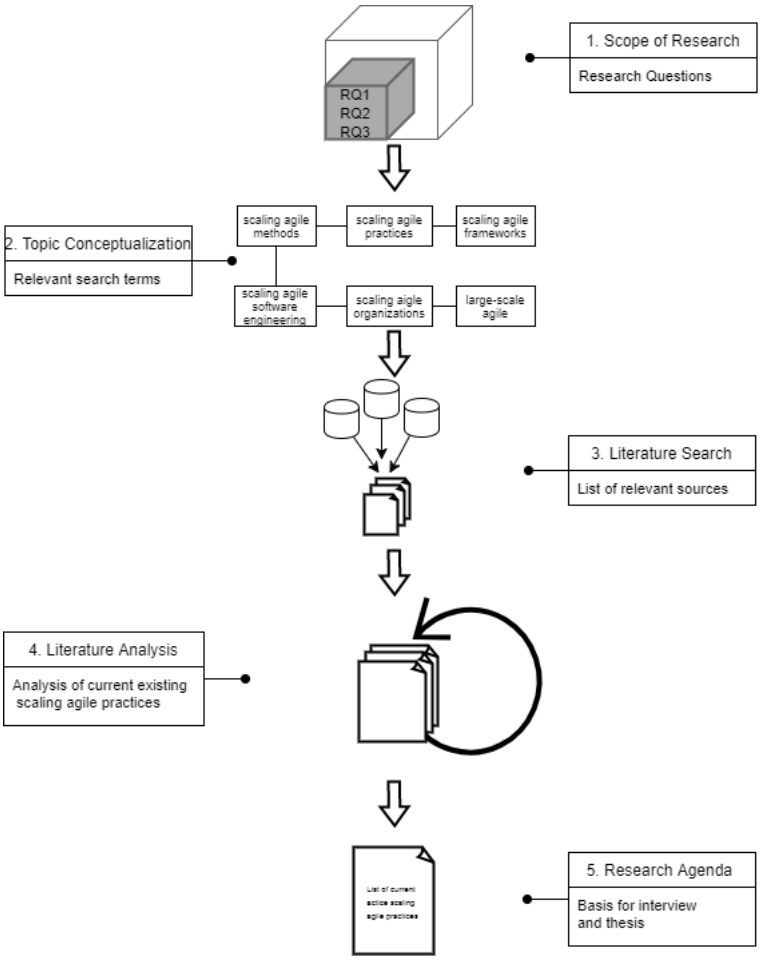


Figure 1.1.: The literature review research process used

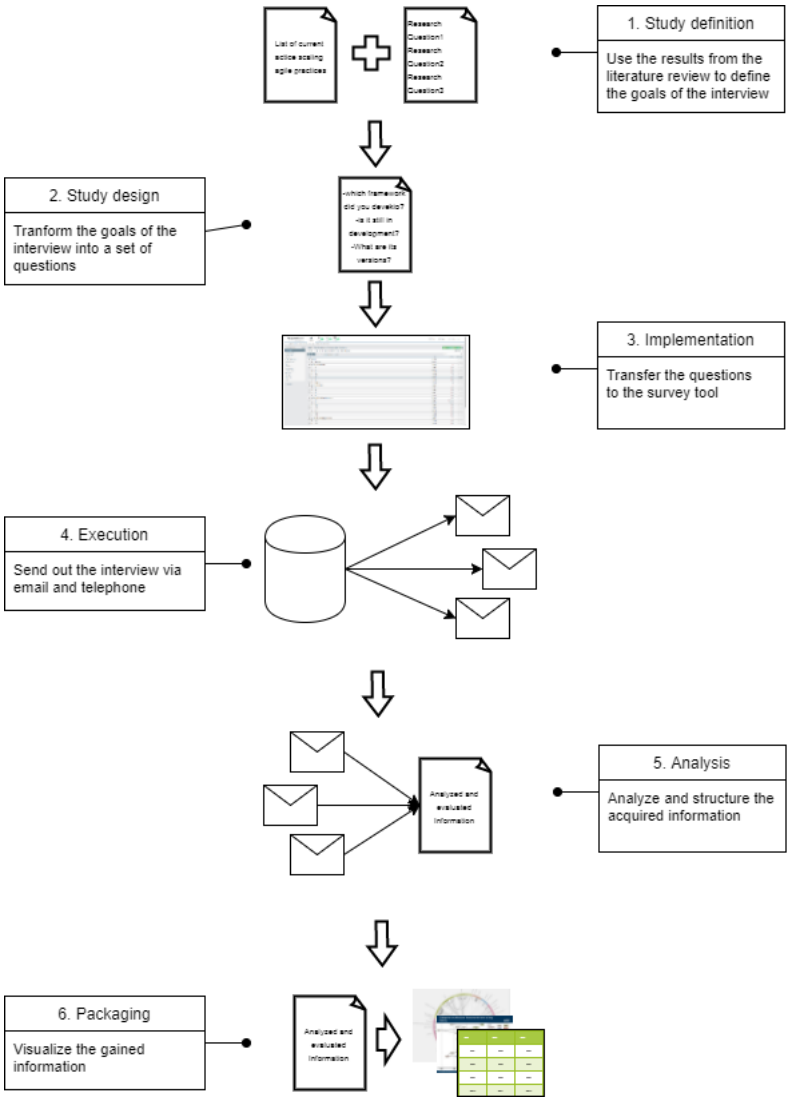


Figure 1.2.: The interview research process used

Part II.

Foundations

2. Foundations

In this chapter we provide an overview over the essential concepts which are utilized in this thesis, to provide a foundation for the Thesis. First, we will consult the agile manifesto to grasp the concept of agility in software engineering. Secondly we will detail the most common agile and lean methods: Scrum, Extreme Programming and Kanban, which often serve as a basis for scaling agile practices and are therefore of vital importance. Finally, we will talk about scaling-up those methods and up-scaled software development and what is generally understood by scaling and the benefits and scaling of it.

2.1. Agile Practices

As Scott W. Ambler, Chief Methodologist/Agile with IBM Rational states: There is no official definition for agile software development and there likely never will be. However, there are multiple unofficial definitions used to describe agile software development, for example:

"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." [11]

However the definition, agile software development is based on the four values and twelve principles of the agile manifesto, which was released in 2001 with the emergance of the Agile Software Development Alliance. These four core values of agile software development are: [24]

"Individuals and interactions over processes and tools"

"Working software over comprehensice documentation"

"Customer collaboration over contract negotiation"

"Responding to change over following a plan"

While they give some ideals more value in this comparison, they do acknowledge that their opposites do also hold value to the development process. Following those four core values, the agile manifesto presents the following twelve principles, to help bring agility into the development process: [24]

> "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."
>
> "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage."
>
> "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale."
>
> "Business people and developers must work together daily throughout the project."
>
> "Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done."
>
> "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."
>
> "Working software is the primary measure of progress."
>
> "Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely."
>
> "Continuous attention to technical excellence and good design enhances agility."
>
> "Simplicity–the art of maximizing the amount of work not done–is essential."
>
> "The best architectures, requirements, and designs emerge from self-organizing teams."
>
> "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."

Agile software development methods therefore differ greatly from more "traditional" methods, are focused on working together with the customer, on delivering working software in short intervals and offer greater flexibility.[10] Using agile methods, organizations are finding that agile project teams, when compared to traditional project teams, enjoy higher success rates, deliver higher quality, have greater levels of stakeholder satisfaction, provide better return on investment (ROI) and deliver systems to market sooner.[85]

2.1.1. Scrum

Being one of the two most commonly used agile methods, Scrum is based on the four core values and twelve principles of the agile manifesto.[28] Scrum is a process for incrementally building software in complex environments with a small team consisting of around seven members, one of wich is the Scrum Master and a Product Owner responsible for the delivered product.[65][72] The Scrum team is self-managing, cross-functional and ideally co-located to facilitate informal communication.[71]
Scrum consists of the Sprint, which is the heart of Scrum, which in itself consists of the Sprint Planning, Daily Scrums, the development work, the Sprint Review and the Sprint Retrospective.[73] These activities utilize the agile artifacts of the Product Backlog, the Sprint Backlog and produce the Product increment, as seen in Figure 2.1.

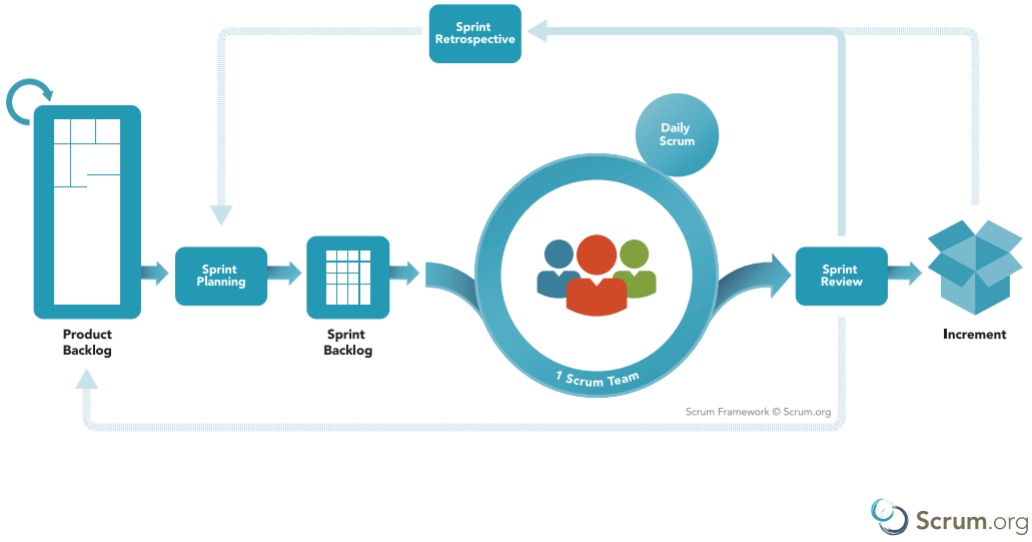


Figure 2.1.: The Scrum Process [74]

Roles

- **Product Owner:**
  The Product Owner is responsible for maximizing the value of the product and the work the Scrum Team delivers. He is the only person responsible for managing the Product Backlog.[73]

- **Scrum Master:**
  The Scrum Master is part of the Scrum Team and serves as a servant-leader for the team. He is responsible for ensuring that Scrum is understodd and enacted, helping the team to get rid of impediments and misunderstandings.[73]

Scrum Events

- **Sprint Planning:**
  Each Sprint starts with a Sprint Planning Meeting, limited to eight hours, where the work to be done in the next Sprint (2-4 weeks) is discussed and items from the product backlog are selected by the entire team to be developed in the Sprint.[73]

- **Daily Scrum:**
  At the beginning of each day, the 15 minute long Daily Scrum is held. Here the Team synchronize activities and create a plan for the day. To achieve this, the work of the previous day is inspected and impediments discussed.[36]

- **Sprint Review:**
  The Sprint Review is held at the end of each Sprint and takes a maximum of four hours. Its purpose is to, together with key stakeholders invited by the product owner,

demonstrate and inspect the Product Increment created by the team in the Sprint and adapt the Product Backlog if needed.[73]

- **Sprint Retrospective:**
  As seen in Figure 2.1, the Sprint Retrospective is held between the Sprint Review and the next Sprint Planning. It is an opportunity for the Scrum Team to analyse itself and the last sprint and to plan improvements for the next Sprint.[36]

Scrum Artifacts

- **Product Backlog:**
  Consisting normally of User Stories, the Product Backlog is an ordered list of everything needed in the product and is the single source of requirements for the envisioned product. It is never complete and changes throughout the development process.[73]

- **Sprint Backlog:**
  The Sprint Backlog contains all the elemts of the Product Backlog that are necessary to meet the agreed upron Sprint Goal. It can be changed by the Development Team during the Sprint, if items in it are deemed unncessary or new items need to be added.[72]

- **Product Increment:**
  Wether or not the Product Owner decides to release it, the Product Increment is a working and releasable part of the envisioned product.[36]

2.1.2. Extreme Programming

As one of the two most commonly used agile methods, Extreme Programming (XP) aims to reduce the cost of change in software development.[28] Extreme Programming is designed to be applied to projects that can be built by teams of two to ten programmers, where a reasonable job of executing tests can be done in a fraction of a day.[14] As seen in Figure 2.2, the XP development process starts with a short overall analysis of the envisioned product, defining what it could do and what it should do first, resulting in a list of Unfinished Features.[87] In the initial analysis, the product is described in the form of business oriented, testable and estimable stories, which can be thought of as the amount of a use case that will fit on an index card.[13] In the next step, the customer chooses the most important features, guided by the development team and their effort estimations of the stories.[87] Thereafter the iterative one week development cycle begins by the customer selecting the stories to be developed, again guided in his descision by the development team, their speed and their estimations.[13] The most distinct features of XP are:[13]

- **On-site customer:**
  A customer sits with the team full-time.

- **Pair programming:**
  All production code is written by two people at one screen/keyboar/mouse.

- **Tests:**
  In Extreme Programming the Test Driven Development approach is followed, requiring each developer to first create unit tests and to then write code fulfilling those tests.

- **Just rules:**
  Extreme programming is an evolving process, acknowleding that every team and project is different. Therefore giving each team the freedom to change certain aspects of XP, to tailor it to their specific situation.

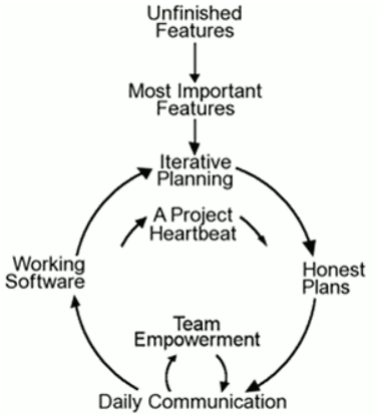


Figure 2.2.: The Extreme Programming Process [87]

2.1.3. Kanban

Kanban is a Japanese word meaning a signboard, and it stems from manufacturing, where it was and is used as a scheduling system. It is a flow control mechanism for pull-driven Just-In-Time production/ development with a focus on the reduction of waste. The basic idea behind Kanban usage is to execute the Lean thinking principles in practice.[5]
The seven principles of lean, adapted to software development by Mary and Tom Poppendieck[59] that Kanban is based on are the following:

- **Eliminate Waste**
- **Build Quality In**
- **Create Knowledge**
- **Defer Commitment**
- **Deliver Fast**
- **Respect People**
- **Optimize the Whole**

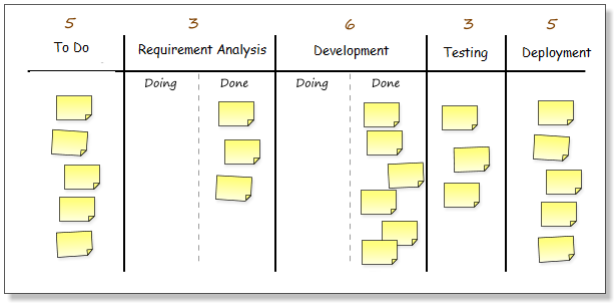


Figure 2.3.: The Kanban Board [88]

Based on these principles, Kanban provides a process to visualize the workflow, limit work in progress (WIP), measure and manage flow, make process policies explicit and to improve collaboratively.[5] This is done using the center piece of Kanban, the Kanban board (Figure 2.3).
The Kanban board is used to map the value stream with columns for each step in the value stream. Kanban cards (tokens for a piece of work) are placed in the columns, representing the current state of the task. When the piece of work meets specified policies, the Kanban card is placed in the next column, therefore visualizing the workflow. The important element of the Kanban board is that work in any column (step in the value stream) is limited, limiting WIP. Therefore, the entire system contains a limited amount of work.[60]

2.2. Large-Scale Agile Development

As defined by Dingsoyr et al.[27], Large-Scale Agile Development refers to 2-9 teams working together on one product in an agile manner. As all the "traditional" Agile Practices we described in the last section have in common that they are designed for one small, colocated team working alone on one product, different - Large-Scale - Agile Practices need to be developed. These Scaling Agile Practices focus on keeping the benefits of Agile Practices of higher velocities in development projects, shorter time to market for new products and a higher software quality, while making them apply to every size of project.[81][85]

2.2.1. Challenges in Large-Scale Agile Development

There are however challenges with Large-Scale Agile Development. Dikert et al.[26] states the main categories of challenges in Large Scale Agile Development as agile being difficult to implement, integrating non-development functions, change resistance, and requirements engineering challenges. Adding to the challenge of agile being difficult to implement, Paasivaara et al.[57] mentiones that in particular, the project they monitored did not succeed in attaining a real agile mindset, and did not adopt all important practices suggested by the framework. Inter-team coordination was insufficient, and the experienced time pressure led to practices not being taught properly, being skipped, or implemented poorly. Boehm et al.[17] identified development process conflicts, business process conflicts and people conflicts as the main categories of challenges.

Part III.

Existing Frameworks

3. Existing Frameworks

This chapter lists all the existing scaling agile practices available at the time of writing alphabetically. We describe each practice and therefore provide a short overview over the current market. Each practice will have its own subsection containing all the relevant information about it. With this we also answer our first research question "What scaling agile practices exist?".

3.1. Agile Software Solution Framework (ASSF)

The Agile Software Solution Framework (ASSF) is a method-independent agile scaling framework, to aid construct high quality processes for the development of large and complex applications [63]. As seen in Figure 3.1 it can be divided into the agile conceptual aspect model (acam), linked to the business via the Business-Agile Alignment Bridge and tools. The agile conceptual model incorporates the aspects of: [64]

- **Knowledge:** The Knowledge Cell is used to engineer and manage all knowledge that is related to agile software development, such as knowledge regarding different process fragments and processes, created by people cooperating and communicating in the organization. It captures and manages this knowledge, to improve performance, increase learning and to aid the decision making process.
- **Governance:** This Cell introduces an integrated agile IT governance as defined by Qumer [30]. It is based on the following five key perspectives of governance for agile processes: lightweight, collaborative, communication-oriented, economical and evolving.
- **Method Core:** Combined with the abstraction element of ASSF, this cell represents the six aspects of an agile software development methodology: agility, process, people, prodict, tools and abstraction. It can be used to link a situational method engineering approach, a feedback mechanism and a meta-model into a software development methodology/process(es).

This agile conceptual model is linked to the Business via the Business-Agile Alignment Bridge, which aligns business goals and agile software development goals, to help foster business value. The Business Cell that the Business-Agile Alignment Bridge connects to the agile conceptual model represents the organization implementing ASSF. It includes the policies, strategies, business goals and its culture, as they have a big, undeniable impact on the execution of any agile software development method [64]. The afore mentioned tools of ASSF are an agile Toolkit, to facilitate the construction and evaluation of processes for the development of complex projects, and a four-dimensional Analytical Tool (4-DAT), as developed by Qumer and Henderson-Sellers [29], for the assessment and analysis of the constructed processes.

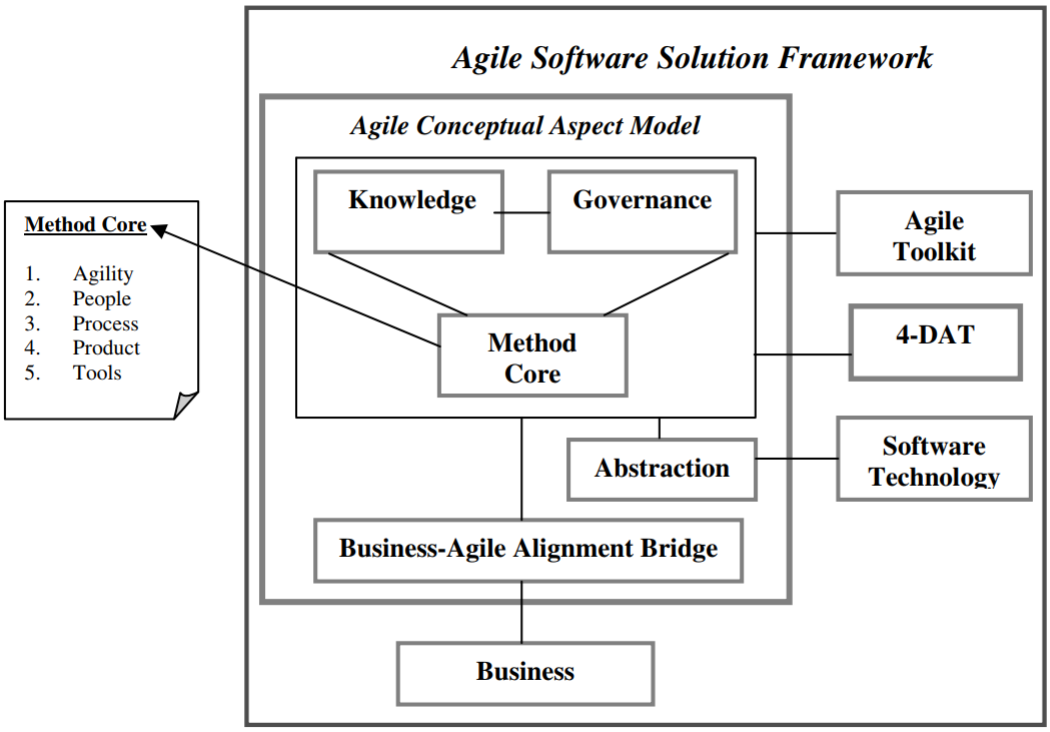


Figure 3.1.: The Agile Software Solution Framework (ASSF) [64]

3.2. Crystal Family (Crystal)

The Crystal Family is a set of customizable development methodologies for small to very large teams. The choice of Crystal Family members is dependent on the size of the project team and the criticality of the project [16]. This choice leads to the two dimensional descision matrix shown in Figure 3.2, where the x-axis represents the size of the team working on the project and the y-axis representing the criticality of the operation. As the criticality and the amount of team members increases, the opacity of the assigned color grows. The criticality of the projects represents their potential for causing damage. The lowest damage being a loss of comfort (C), then the loss of discretionary money (D), after that the loss of essential money (E) and finally the potential loss of life (L). The size of the project teams and therefore the x-axis is sectioned into teams of up to 6, 20, 40, 100, 200, 500 and over 1000 people. Not every combination of sections (C6,D20,etc.) has its own methodology in the Crystal Family, but the Crystal Methodologies are rather used in a wide field and combined and adapted to the specific requirements, as their creator believes there is no "one-size-fits-all" development process. However, they all focus on a development cycle no longer than three months and personal, unofficial communication with people on-site. The two most defined Methods are Crystal Clear and Crystal Orange.

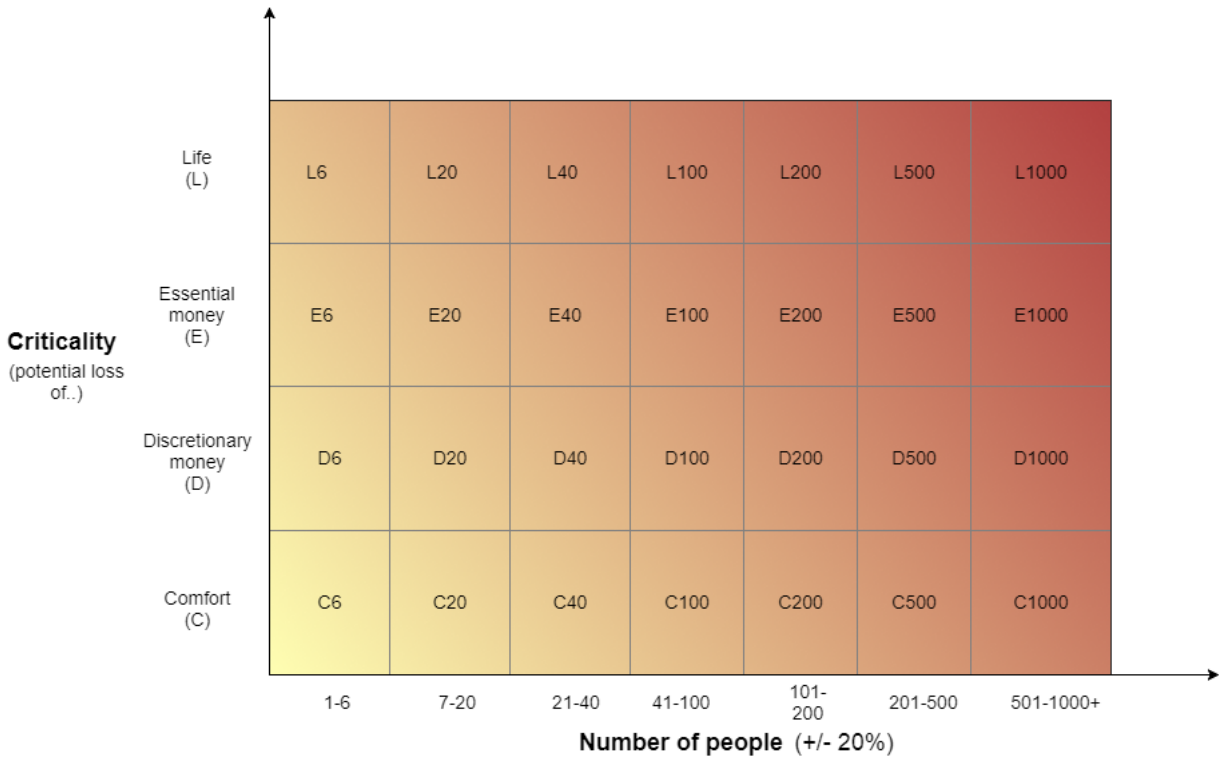


Figure 3.2.: The Crystal Family Descision Matrix [16]

3.2.1. Crystal Clear

Crystal Clear is the smallest methodology in the Crystal family. It is tailored to a D6 project, but can also be applied to projects in the size and criticality of C6 or E6. It can possibly be extended from six to ten people, applying to C10,D10 and E10 [16], as long as they sit in the same or adjacent rooms and all have access to close communication with each other. A. Cockburn, the creator of the Crystal family of methodologies, describes Crystal Clear as follows:

"The lead designer and two to seven other developers in a large room or adjacent rooms, with information radiators such as whiteboards and flip charts on the wall, having access to key users, distractions kept away, delivering running, tested, usable code every month or two (okay, three at the most), periodically reflecting and adjusting on their working style." [22]

Boehm [16] lists the elements of Crystal Clear along these lines:

Documents and artifacts: release plan, schedule of reviews, informal/low-ceremony use cases, design sketches, unning code, common object model, test cases, and user manual

Roles: project sponsor/customer, senior designer-programmer, designer-programmer, and user(part time at least)

Process: incremental delivery, releases less than two to three months, some automated testing, direct user involvement, two user reviews per release, and methodology-tuning retrospectives. Progress is tracked by software delivered or major decisions reached, not by documents completed.

3.2.2. Crystal Orange

The Crystal Orange model is mainly targeted at a D40 project. It is aimed at 20-40 programmers working together in the same building on a project with medium risk, where the time to market is of big importance and the duration is estimated between one and two years [21]. The elements of the Crystal Orange model are described as follows: [16]

Documents & artifacts: requirements document, release plan, schedule, status reports, UI design document, inter-team specs, running code, common object model, test cases, migration code, and user manual

Roles: project sponsor, business expert, usage expert, technical facilitator, business analyst, project manager, architect, design mentor, lead desinger- programmer, designer-programmer, UI designer, reuse point, writer, and tester

Process: incremental delivery, releases less than three to four months, some automated testing, direct user involvement, two user reviews per release, and methodology-tuning retrospectives.

3.3. Disciplined Agile 2.0 (DA 2.0)

The Disciplined Agile Framework provides frameworks for certain levels of a company. It supplements core agile practices with more disciplined approach, extending the construction-focused lifecycle, adds modeling, documentation and government strategies [12]. The official definition of Disciplined Agile reads:

"The Disciplined Agile (DA) process decision framework provides light-weight guidance to help organizations streamline their processes in a context-sensitive manner, providing a solid foundation for business agility. It does this by showing how the various activities such as Solution Delivery, IT Operations, Enterprise Architecture, Portfolio Management, Finance, Procurement and many others work together. The framework also describes what these activities should address, provides a range of options for doing so, and describes the tradeoffs associated with each option"[43]

The four levels of the Disciplined Agile framework as seen in Figure 3.3 are described as follows: [43]

Disciplined Agile Delivery(DAD) DAD addresses all aspects of solution delivery from beginning to end, in a streamlined manner. This includes initial modelling and planning, forming the team, securing funding, continuous architecture, continuous testing, continuous development, and governance all the way through the lifecycle. The framework includes support for multiple delivery lifecycles, including but not limited to a basic/agile lifecycle based on Scrum, a lean lifecycle based on Kanban, and a modern agile lifecycle for continuous delivery.

Disciplined DevOps Disciplined DevOps is the streamlining of IT solution development and IT operations activities, and supporting enterprise-IT activities, to provide more effective outcomes to an organization.

Disciplined Agile IT (DAIT) As the name suggests DAIT addresses how to apply agile and lean strategies to all aspects of IT. This includes IT-level activities such as enterprise architecture, data management, portfolio management, IT governance, and other capabilities.

Disciplined Agile Enterprise A Disciplined Agile Enterprise (DAE) is able to anticipate and respond swiftly to changes in the marketplace. It does this through an organizational culture and structure that facilitates change within the context of the situation that it faces. Such organizations require a learning mindset in the mainstream business and underlying lean and agile processes to drive innovation.

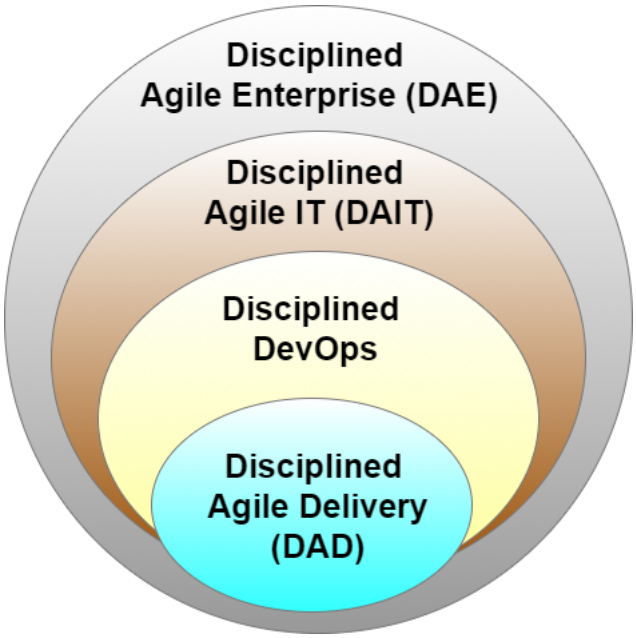


Figure 3.3.: The scope of the Disciplined Agile 2.0 framework[43]

3.4. Dynamic Systems Development Method Agile Project Framework for Scrum (DSDM)

The Dynamic Systems Development Method Framework (DSDM) focusses on people instead of tools. It focusses on understanding the needs of a business and delivering working software solutions. It provides a framework of controls and best practices for agile software development [78]. The philosophy behind the Dynamic Systems Development Method framework is best described by the following: [77]

- Development is a team effort. It must combine the customers' knowledge of the business requirements with the technical skills of IT professionals.
- High quality demands fitness for purpose as well as technical robustness.
- Development can be incremental - not everything has to be delivered at once, and delivering something earlier is often more valuable than delivering everything later.
- The law of diminishing returns applies - resources must be spent developing the features of most value to the business.

DSDM flips the traditional view on a project, where the features of the product (the requirements) are the only fixed point, which leads to increasing cost and time, to a view where time and cost are fixed and any adjustments are made on the features of the product, in full agreement of all parties involved [79]. It follows these eight principles: [42]

1. **Focus on the Business Need:** Every decision taken during a project should be viewed in the light of the overriding project goal - to deliver what the business needs to be

delivered, when itneeds to be delivered. It is important to remember that a project is a means to an end, not an end in itself.

2. **Deliver on Time:** In order to fulfil this principle, DSDM brings teams to:
   - Timebox the work
   - Focus on business priorities
   - Always hit deadlines
   - Build confidence through predictable delivery
3. **Collaborate:** Teams that work in a spirit of active cooperation and commitment will always outperform groups of individuals working only in loose association. In order to fulfil this principle, DSDM gets its teams to:
   - Involve the right stakeholders, at the right time, throughout the project
   - Encourage pro-active involvement from the business representatives
   - Ensure that all members of the team are empowered to take decisions on behalf of those they represent
   - Build a one-team culture
4. **Never compromise quality:** In DSDM, the level of quality to be delivered should be agreed at the start. All work should be aimed at achieving that level of quality - no more and no less. This is achieved by constant reviews and appropriate design and documentation.
5. **Build Incrementally from Firm Foundations:** One of the key differentiators for DSDM within the Agile space is the concept of establishing firm foundations for the project before committing to significant development. DSDM advocates first understanding the scope of the business problem to be solved and the proposed solution, but not in such detail that the project becomes paralysed by overly detailed analysis of requirements. This is achieved by carrying out analysis and enough design up front and formal and informal re-assessments throughout the ongoing project.
6. **Develop Iteratively:** The concept of iteration is at the heart of everything developed as part of the DSDM approach. It is very rare that anything is created perfectly first time and it is important to recognise that projects operate within a changing world.
7. **Communicate Continuously and Clearly:** Poor communication is often cited as the biggest single cause of project failure. Therefore DSDM encourages:
   - Informal, face-to-face communication at all levels
   - Daily team stand-up sessions
   - Workshops, with a facilitator where appropriate
   - Visual communication practices such as Modelling and Prototyping
   - Demonstrating the Evolving Solution early and often
   - Keeping documentation lean and timely

- Managing the expectations of the stakeholder at all levels throughout the project

- Always aiming for honesty and transparency in all communication

8. **Demonstrate Control:** It is essential to be in control of a project at all times and to be able to demonstrate that this is the case. This can only be achieved by reference to a plan for the work being done, which is clearly aligned with agreed business objectives. It is also vital to ensure transparency of all work being performed by the team.

3.5. Enterprise Scrum (eScrum)

Enterprise Scrum, as the name suggests, scales Scrum to the enterprise level. It looks at all attributes of Scrum, like sprint length, estimation units, backlog item size, feedback-loop frequency, assigned resources, planning processes and the ranking system, and escalates them to the enterprise level [32].
It therefore creates a Scrum-model of the quarterly enterprise activities as seen in Figure 3.4, where the quarter is equivalent to a Scrum-sprint, including planning, production of deliverable products and a retrospective. It also uses a weekly standup for Product Managers, Scrum Masters and Team Leaders to discuss progress, short-term plans and difficulties [32]. Enterprise Scrum puts a big focus on the iterative quarterly planning. The official definition of Enterprise Scrum is defined as:

"Enterprise Scrum is a customer-centric, co-evolutionary, empirical, subsumption-based management framework to deliver the most business value in the shortest amount of time while balancing the benefits for all people involved, based on an abstraction, generalization, extension and parametrization of Scrum for generic, business, technique-pluggable, and scaled management purposes"[15]

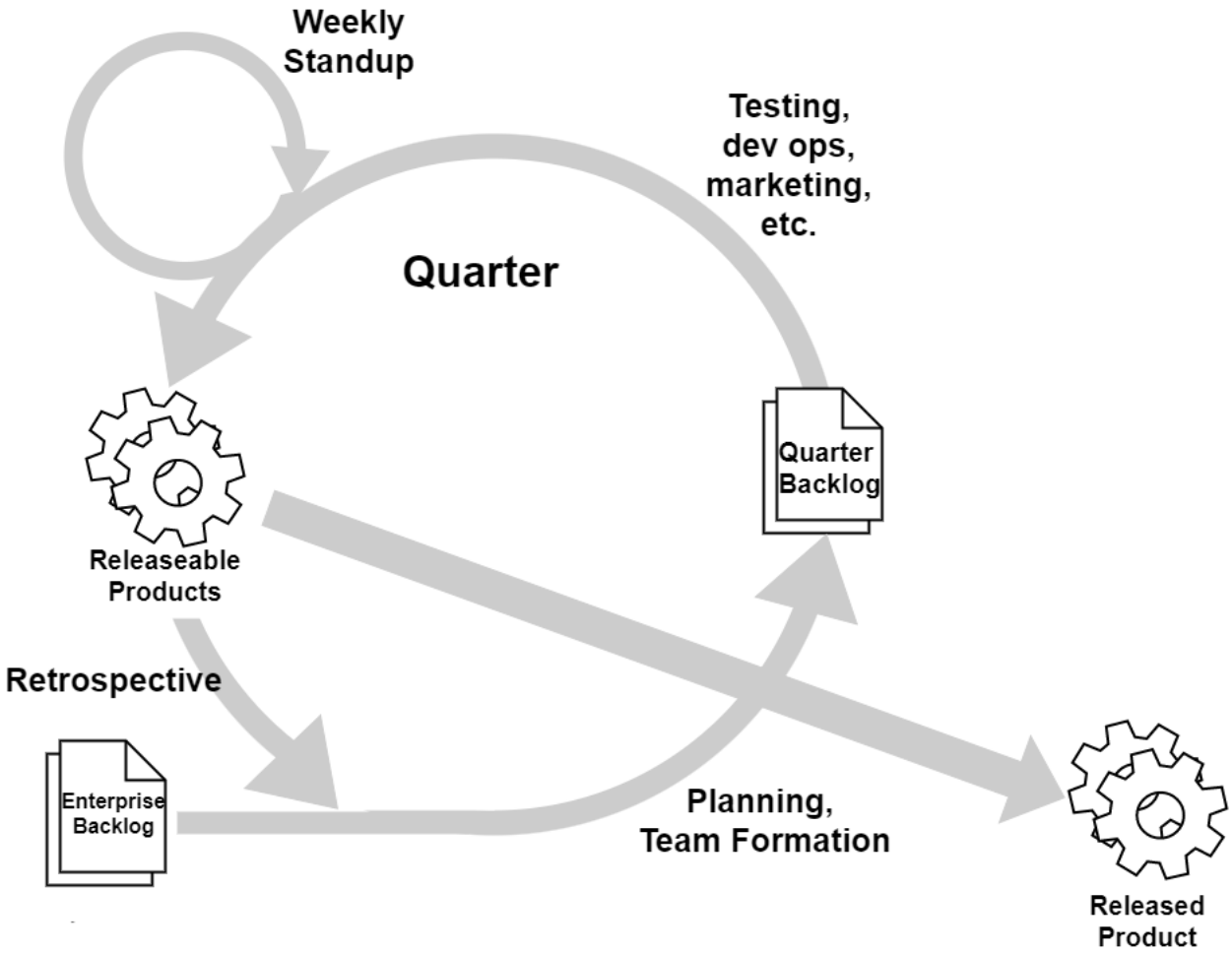


Figure 3.4.: Enterprise Scrum[32]

3.6. Enterprise Transition Framework (ETF)

The Enterprise Transition Framework (ETF) is a Framework by the agile42 company designed to help companies of all sizes to transform their business processes and culture to an agile one. Through a iterative process,the company and each part of the business is first assessed, then a strategy is formed and pilot projects are launched. If those pilot projects succeed, the desired scaling agile process is rolled out to the tested part. The official definition reads:

"The Enterprise Transition Framework is not only a unique approach to enterprise continuous improvement, but also a library of reusable tools and training modules which enable an organization to evolve very rapidly in their own way. ETF practices have proven successful across a wide-range of industries. Through a scientifically validated approach the ETF enables your organization to foster a continuous improvement culture, which will

allow your business strategy to succeed." [3]

3.7. Event Driven Governance

The Event Driven Governance methodology, formerly known as Enterprise Agile Delivery and Agile Governance Practice (EADAGP), focuses on a lean governance framework. The Event Driven Governance framework introduces a governance buffer zone, which aims to protect agile teams from the slowness, friction and rigidity of traditional IT governance [54]. Nevertheless the agile teams are enabled to engage with the IT governance for the escalation of problems, critical decisions and oversight over the bigger picture. The buffer zone is comprised of three tiers, which include top-down governance, community governance and self-governance. Event Driven Governance introduces a new item: the governance backlog. Similar to the product backlog, the governance backlog contains prioritized governance requirements, which are identified at various times in the agile delivery process. This new item is cared for by the new agile governance owner, who is responsible for it [50]. Event Driven Governance is build upon the following ten principles:[51]

- Do not create standing governance bodies, boards or committees.
- Enforce critical enterprise policies when policy events trigger the governance model
- Implement trust-based governance via self-governance and community governance designs
- Focus on the vital few principles and policies that will drive desired business and IT performance
- Use policy enforcement event models (governance processes and events) to define where, when and how policies will be enforced
- Define core policy threads that connect oversight processes, policy enforcement points and policy events into a policy enforcement model
- Explicitly design in self-governance, community governance and trust-based capabilities into your governance model
- Leverage a common Governance Reference Model to define, model, implement and operate your governance model
- Collapse key IT governance processes and mechanisms into the fewest number of policy events that will meet Enterprise Governance requirements
- Design the governance model to be Just-In-Time and Just Enough, and no more than that

3.8. eXponential Simple Continuous Autonomous Learning Ecosystem (XSCALE)

The eXponential Simple Continuous Autonomous Learning Ecosystem (XSCALE) extends the agile manifestos' principles to the product and portfolio level and changes the companies culture and decicion making. It aims at tranforming a company into an agile organization not by scaling Agile to the company, but by descaling the organization of the company to Agile[53]. The six principles of XSCALE are what creates the name:

- **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

As XSCALE sees itself as the next step of the agile manifesto, its values are formulated in the same manner, where they value certain things over others, but do not deprive those other values of their importance. They value: [53]

- Individuals and interactions over processses and tools
- Business throughput over comprehensive accounting
- Customer collaboration over contract negotiation
- Responding to change over following a plan
- Decentralized ownership over delegating control

3.9. FAST Agile

"The goal of FAST Agile is to gain control of your project and achieve hyper-productivity by TRUSTING your developers to do the right thing at the right time and letting them SELF-ORGANIZE." [62] This official description of the FAST Agile process shows the two major focusses of FAST: self organization and the trust that goes with it. The FAST Agile process congregates all your working teams into clans in a tribe. Each tribe (and therefore the number of people working on a project) is limited to Dunbar's number: 150. It is a suggested cognitive limit to the number of people with whom one can maintain stable social relationships. FAST is heavily influenced by Open Space Technology (OST)[56], as such it demands the co-location of the entire development team. In FAST, there is one big meeting, called the FAST meeting, held at the start of every iteration. It is comprised of the following five steps:[62]

I. **Show and Tell**
During the first phase, each team that has completed work from the previous iteration will show their work to everybody. After everyone has had the chance to answer questions, the Product Director/ Manager and Stakeholders will give critique and feedback. The Product Director will at this point be able to make refinements and add or drop features. This is also the point where he declares stories to be "done".

II. **Marketplace**
Phase two of the FAST Agile process is copied from Open Space Technology, as this is the phase where volunteers will approach the stage and announce to everyone what story or problem they would like to work on in the next iteration. After all volunteers have assigned their work to a room, the Product Director inspects the marketplace and may ask questions/ use his veto right.

III. **Announcements and Alignment of Vision**
In the thrid phase, while everyone is present, any public announcements ought to be made and the vision for the product should be reiterated.

IV. **Team huddle, dependency management and planning**
The fourth phase consists of the story shepherds (the volunteers from phase II) reviewing the marketplace and identifying any dependencies they need to be aware of. Afterwards they go to their respective rooms, waiting for the people that want to join them for this iteration on the task they have presented in phase II.

V. **Get coding!**
The last phase is not really a phase of a meeting but the start of the iteration, in which the newly formed clans work on their tasks.

3.10. Holistic Software Development (HSD)

Holistic Software Development combines Lean Portfolio, Programme and Project Management practices with agility, complex software practices and soft practices into the H-Model (Figure 3.5), which is derived from the old V-model.[46]

As you can see in Figure 3.5, the H-model is divided into three categories:

I. **Portfolio and Programme Management:**
The left side of the H-model represents the portfolio and programme management view. It focusses on architecture, forming the right teams and managing requirements.

II. **Release Planning:**
The cross-bar of the H model concentrates on agile development, joining up strategy and delivery teams and de-conflicting agile delivery and management approaches.

III. **Product Delivery:**
As with the classical V-model, the right side focusses on quality management, product delivery and integration, using feedback cycles and a definition of done.

Figure 3.5.: The H-model used by HSD [45]

The big differences to the old V-model are that the H-model is used iteratively, not focusing on vertical decomposition and recomposition, but on releasing product increments through iterative processes. HSD can be used with a variety of different agile and lean software development methods like Scrum, XP or Kanban.
Figure 3.6 provides a more detailed description of the H-model, called "the big picture". It shows the roles and artifacts used in HSD, as well as the project stream, going from Customer Requests and Backlog refinement, through Product Development, to Systems and Acceptance tests.

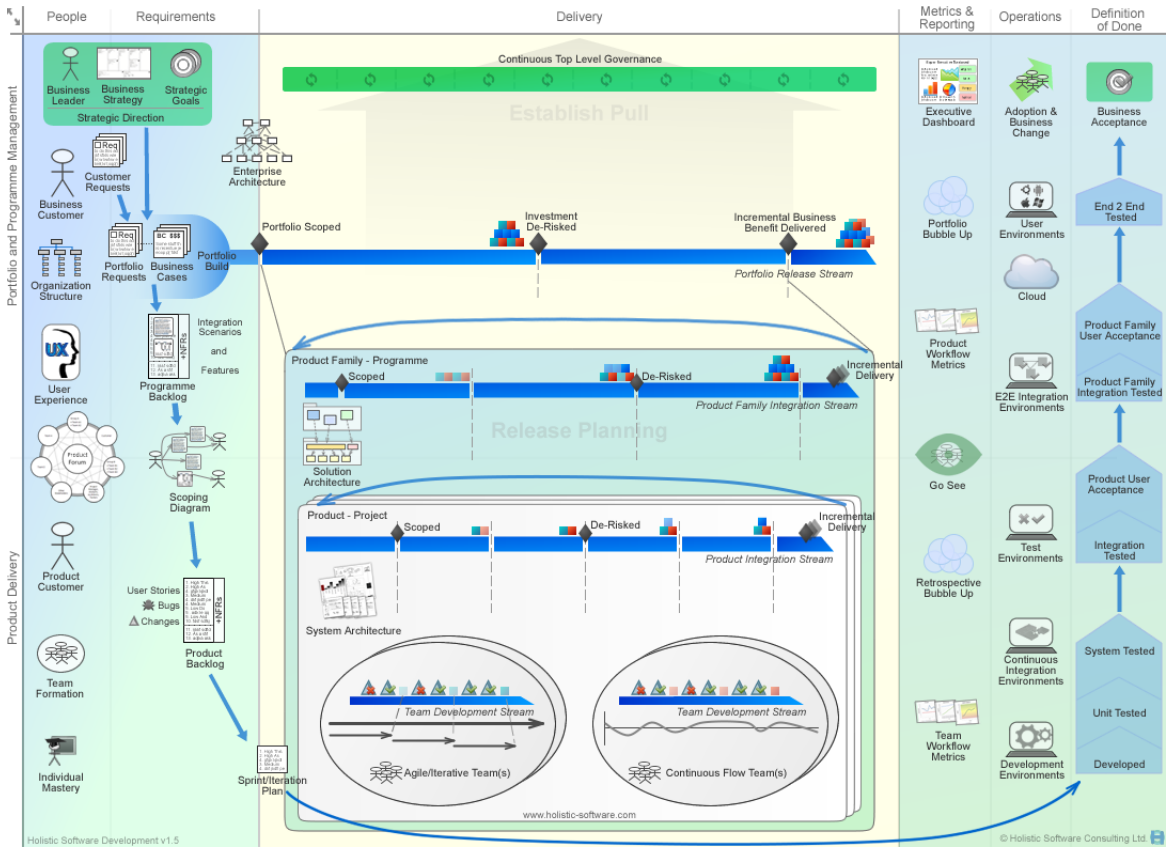


Figure 3.6.: The "Big Picture" [47]

3.11. Large Scale Scrum (LeSS)

The Large Scale Scrum Framework is a scaled-up version of the normal one team scrum. It tries to be as close to one team scrum as possible by introducing the least amount of new roles, artifacts and processes [40]. LeSS divides the work process into sprints, like normal scrum. In those sprints however, multiple teams work on the same product, albeit on different product backlog items. The whole project therefore has the same product backlog and the same product owner (PO). LeSS defines the role of the product owner as having the same responsibilities as in regular one-team classic scrum: owning the vision for the total product portfolio, business plan, road map and dates. The focus of the product owner is return on investment [41].
As depicted in Figure 3.7, each sprint starts with a sprint planning meeting 1, in which the PO and representatives of all teams work out the next items to work on. Thereafter the sprint planning meeting 2 is held for each team independantly and in parallel, which resembles the normal scrum sprint planning meeting. Afterwards begins the normal scrum process of daily sprints and product backlog refinement, which can optionally be held with representatives of all teams. After the sprint is completed, a sprint review is held with the product owner, members of all teams and relevant customers/users and other stakeholders. It is recommended that this is held in the form of a "science fair": a large room with multiple areas for the teams to present and discuss their developed items. Finally an overall retrospective is held to improve the overall system. It is limited to 45 minutes per week of sprint and includes the PO, scrum masters and rotating representatives of each team[20]. For projects incorporating more than eight teams the LeSS Huge framework was developed.
It is an alteration of the normal LeSS framework accounting for the difficulties that arise from hundreds of people working together. LeSS Huge splits the project into different requirement areas, each representing a normal LeSS unit [41]. It therefore adds the new role of area product owner (APO) and the product owner team, which is comprised of all area product owners and the product owner. The area product owner is the product owner of an area backlog, which is a view onto the product backlog for one area.

"For large groups, LeSS hits the sweet spot between defined concrete elements and empirical process control."[20]

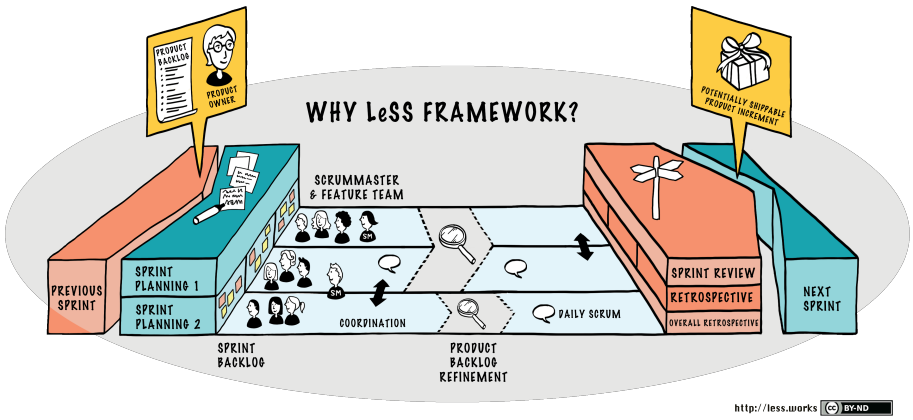


Figure 3.7.: The LeSS framework [20]

3.12. Lean Enterprise Agile Framework (LEAF)

The Lean Enterprise Agile Framework (LEAF) is a scaling agile practice developed by Satisha Venkataramaiah. It is a systems thinking tool for helping oragnizational agility [44]. LEAF aims to help organizations find the constraints that are limiting their agile scaling potential [84]. The Lean Enterprise Agile Framework is based on the following four principles:

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

These four principles are used to continuously break constraints in a five step process. Those constraints are then being changed by protecting or improving, so that they are no longer a constraint to the scaling. This process is supposed to be run constantly to iteratively remove all constraints [44].

3.13. Matrix of Services (Maxos)

The Matrix of Services (Maxos) is supposedly used by companies in the silicon valley to get an advantage over the competition. [76]
Formerly known as the Continuous Agile Framework (CAF), the Matrix of Services is build on the principle of ”divide and conquer”. It divides each product into tens, even hundreds of small services. As seen in Figure 3.8, each service is build, tested and operated by a small team, which is responsible for multiple services [76].
Maxos focusses on:[75]

- Continuous Delivery

- Frequent releases
- Developer responsibility
- Feature gates
- Delivering user experience instead of requirements

The Matrix of Services is simple to scale: as you add more to your product you add more services. More services lead to more teams, if the number of services is higher than what the current number of teams can work on. This leads to the need of teams being truly multifunctional. Maxos relies on coordination and communication without big meetings and self organizing teams. It bestows responsibility onto the developer by making Quality Assessment into a consulting role.

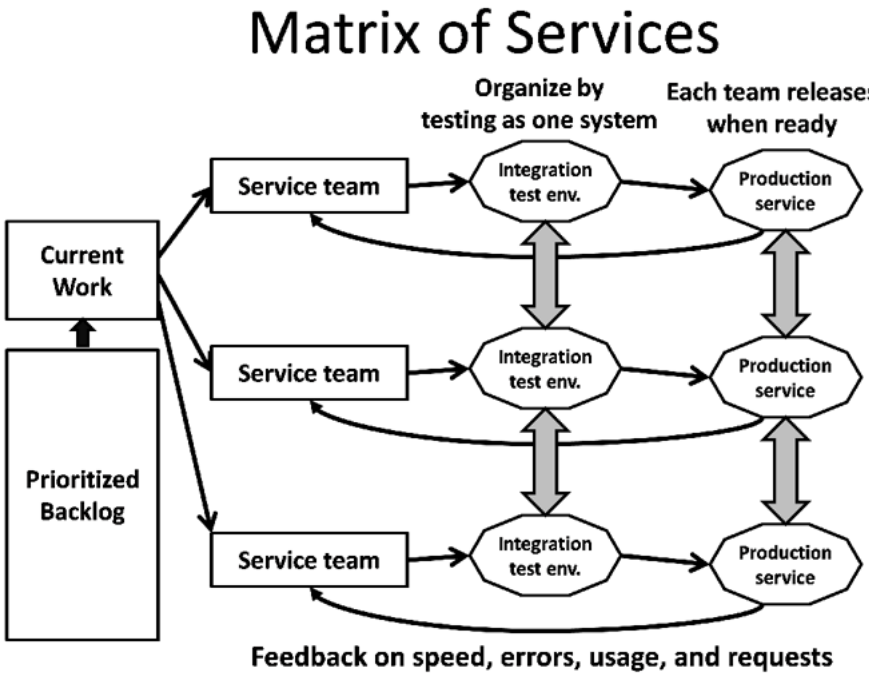


Figure 3.8.: The Matrix of Services [76]

3.14. Mega Framework

The Mega Framework adds an additional layer to Scrum. Mega is a set of practices and meetings that provides synchronization to all levels: from the stakeholders over the management to the team. The teams in Mega are focused on features instead of software components. It uses one big Mega backlog, whose prioritization can change the focus of existing teams.[48] The Mega Framework is comprised of the following eight meetings: [49]

Weekly pre-planning: Held weekly by the product owner and stakeholders to discuss the prioritized backlog items that will be worked on this week

Sprint Planning: The normal sprint planning meeting.

Mega Planning: This meeting is held by the product owner and members of each team. Its agenda is the calendar of upcoming releases and the stories of each of them. It is an opportunity for inter-team communication, where important issues and details are uncovered and discussed.

Mega Standup: In the Mega Standup, each team is invited to discuss if the sprints are on time and the impacts if they are not. This mid-sprint meeting is used to synchronize the teams.

Mega Retrospective: The Mega Retrospective is held every six sprints, to identify global impediments that can not be identified in team retrospectives. The scrum master and one member of every team bring their prioritized impediments to this meeting, to find global ones, common to at least two of them.

Sprint Review: The only difference to normal scrum sprint reviews lies therein that at least one member of each feature team is present in each other teams meetings.

Weekly product owner and scrum master meeting: In this meeting the PO and scrum masters of each feature team meet to discuss impediments, main features planned, changes, pending issues, corporate policies and decisions.

Regular Mega meetings with business area: This monthly meeting is held with all product owners and managers of the operational areas to share the most relevant points that have to be addressed in the feature teams and the business.

3.15. Nexus

”Nexus is a framework consisting of roles, events, artifacts, and techniques that bind and weave 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.”[70]
Nexus scales normal one-team scrum to the level of 100 people. It focusses on dependencies and interoperation between Scrum Teams to accomplish one Integrated Increment every Sprint. As seen in Figure 3.9, Nexus scales scrum by adding an integration team, consisting of a scrum master, the product owner and integration team members. They are responsible for solving impediments and that the product increment is delivered on time.

Therefore they may work in the normal development teams as necessary. Its role is to coach and coordinate the other teams and to administer the correct implementation of the Nexus framework [52].
Nexus also uses a new artifact: the Nexus backlog. It is made up of all the items from the individual team sprint backlog, to highlight dependencies and to give higher transparency. At the start of each daily scrum the Nexus Daily Scrum is held, which representatives of all teams attend to discuss new identified dependencies and to share information across all teams. After each sprint, representatives of all teams have the opportunity to meet and identify issues that have impacted more than a single team in the Nexus Sprint Retrospective. [70]

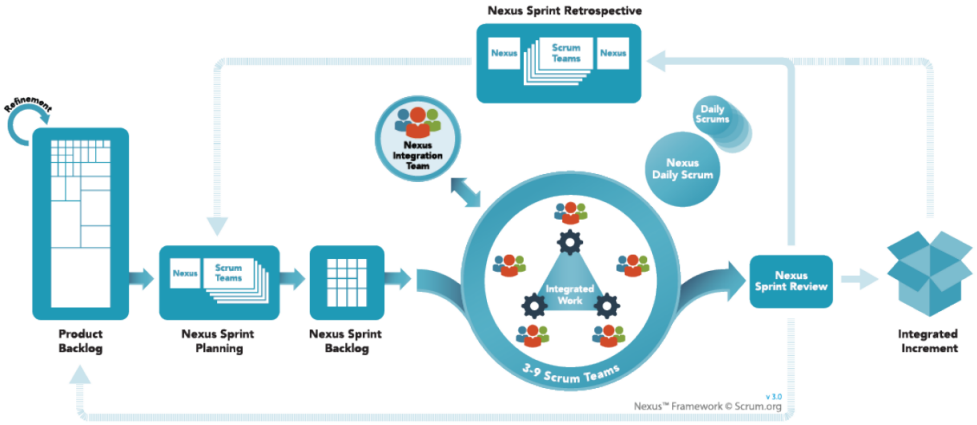


Figure 3.9.: The Nexus Framework [70]

3.16. Recipes for Agile Governance in the Enterprise (RAGE)

RAGE focusses on allowing quick decisions in accordance to lightweight artifacts created using minimum resources. It can be employed in any process like agile, hybrid or plan-driven.[7]
As seen in Figure 3.10, RAGE distinctively divides into three levels: [82]

- **Project level** The project level describes the team level. It utilizes all roles, artifacts and meetings and functions like a normal scrum team. Additionally it is recommended that they make use of burn down charts and taskboards to keep track of their progress.

- **Programm level** At the programm level area product owners and program managers conduct release planning meetings and release reviews, as they focus on product readiness and uphold responsibility for their area of the product.

- **Portfolio level** In RAGE, the portfolio level focusses on decision and planning. Portfolio, area product owners and program managers conduct portfolio grooming meetings to develop the portfolio backlog. This level is used to determine new initiatives, funding decsicions, monitoring and for quality assessment.

As the name suggests, RAGE provides the company with many different "recipes" to solve common problems with scaling (agile) at the different levels, which still allow for customization, while providing enough clear information to have them implemented easily. Those recipes are not only made to be tailored to the specific environments the company may face, but RAGE also provides guidance on how to create new recipes on your own. [82]

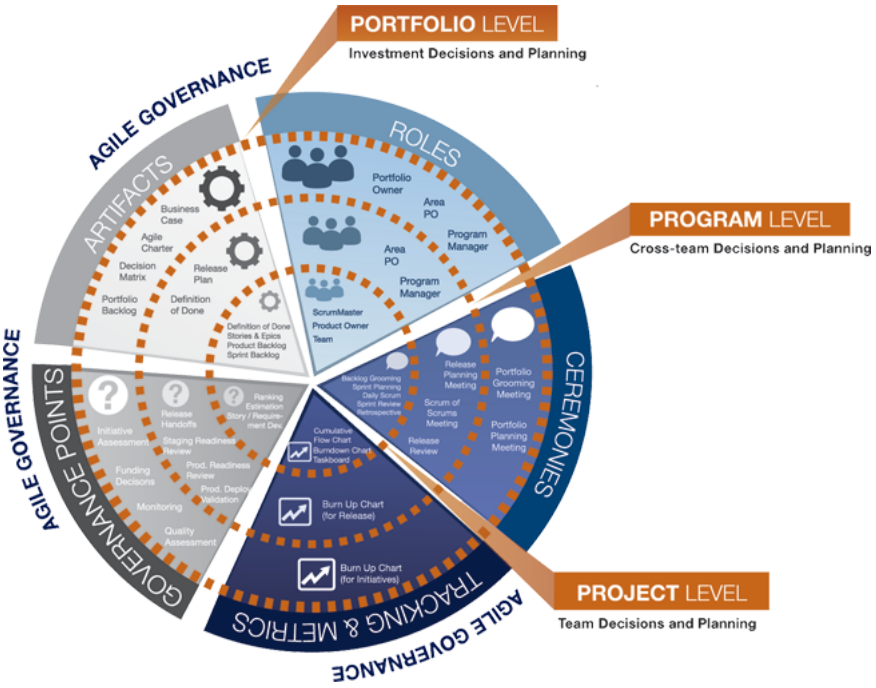


Figure 3.10.: The Recipes for Agile Governance in the Enterprise (RAGE) [23]

3.17. Resilient Agile

"Resilient Agile (RA) is an attempt to put the necessary engineering back into software development without losing the "get to code early" focus that agile gets right. RA is based on a time-tested method (ICONIX/DDT) of developing an initial problem domain model, then decomposing a system into collaborating use cases and elaborating each use case by doing a conceptual models, views, controllers (MVC) decomposition."[33]

Resilient Agile is a process used to scale agile by introducing a minimum amount of planning and by splitting the project down use cae boundaries, to enable parallel work. Resilient Agile implements a complete sunny-day/rainy-day description for all use cases, to add resilience to the software designs. By combining a UML-design approach with extensive prototyping RA is both feedback-driven and plan-driven at the same time, and therefore minimizes planning costs. It hits the sweetspot between over-planning and under-planning. Resilient Agile does however not specify a management process for delivering code, but focuses strictly on the engineering design of software deveopment. It is possible to use RA with two-week scrum sprints. If the sprint/backlog paradigm is desired, it is straightforward to populate the backlog with controllers from the MVC use case decompositions. At last, Resilient Agile attempts to eliminate "ceremony"(e.g. meetings) from the development process. However, RA recognizes the need for organized communication between team members, introducing a 90 minute standup meeting once every week.[69]

3.18. Scaled Agile Framework (SAFe)

The Scaled Agile Framework version 4.5 is a comprehensive framework for scaling agile accross the 4 different levels of an organization, as depicted in Figure 3.11. Each level carries its own activities and all levels are tied together. SAFe integrates Agile and Lean practices at all four levels.[8]
The first of those levels is the team level. It regulates the different agile teams and the development process, which is modeled after scrum, but utilizes XP and Kanban.
Above the team level is the program level, which introduces the concept of the agile release train (ART), comprising of four two-week iterations followed by a three-week Hardening, Innovation and Planning iteration, consisting of 50-124 people.[18]
Teams are dedicated to the ART and synchronously develop and release potentially shippable increments(PSI). To help them accomplish this, the architectural runway is developed, to support the development teams with the needed infrastructure.[83]
The large solution level combines different ART into a solution train, therefore allowing hundreds of people to work together on one large and complex product. This level focusses on capturing requirements in solution intent, the coordination of suppliers, and the need to ensure compliance with regulations and standards.[34]
Above all, the portfolio level focusses on the principles, practices, and roles needed to initiate and govern a set of development Value Streams (solution trains). "This is where strategy and investment funding are defined for value streams and their Solutions. This level also provides Agile program guidance and Lean governance for the people and resources needed to deliver solutions."[35]

These four levels, as of SAFe version 4.5, can be combined in four different ways to cater to the specific needs of different organizations:[34]

I. **Essential SAFe:**
Essential SAFe consists of the team and programm levels. It is the heart of SAFe and the starting point for all organizations looking to scale agile using SAFe.

II. **Portfolio SAFe:**
Portfolio SAFe adds the portfolio level to essential SAFe and is suggested for start-up like companies of up to 125 employees working on the same product, as it provides portfolio strategy and investment funding.

III. **Large Solution SAFe:**
Large Solution SAFe is designed for enterprises that are building large and complex solutions, but do not require expertise at the portfolio level, like aerospace, defense and government organizations.

IV. **Full SAFe:**
Last but not least, Full SAFe supports building large, integrated solutions that require hundreds of people or more. It is used for scaling the whole organization, from the team up to the portfolio level.

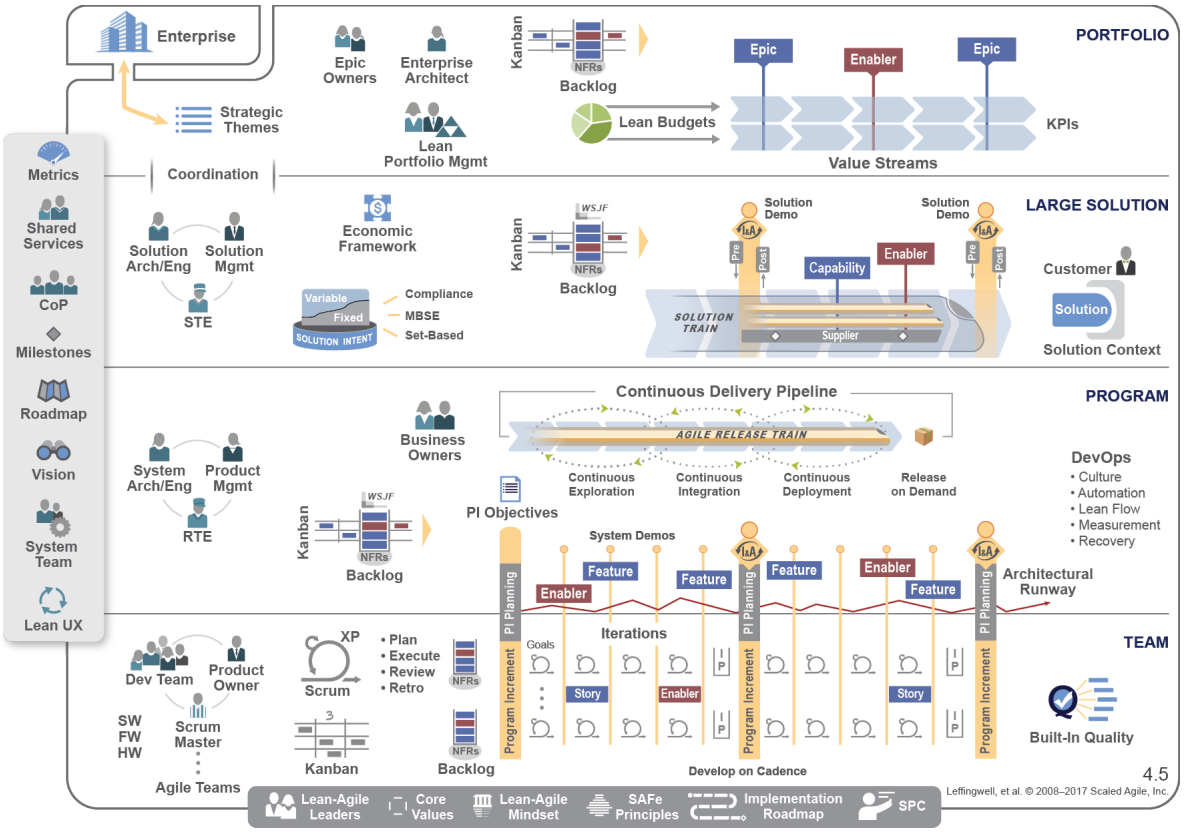


Figure 3.11.: The Scaled Agile Framework 4.5 [34]

3.19. ScALeD Agile Lean Development (ScALeD)

Scaled Agile Lean Development is a set of principles and values that aims to scale agile by changing the culture of the enterprise. ScALeD focusses on trying to incorporate the employees into the process early, on teaching management the fundamentals of an agile culture and on reflecting over processes and practices.[2]
To achieve this, ScALeD introduces thirteen principles, grouped into five categories:[1]

I. **Excited customers:**
   - Define value and how it is created
   - Produce small, deliverable increments

II. **Happy and productive employees:**
   - Create independent, cross-functional teams
   - Authorize and empower your employees

III. **Global optimization:**
   - Create transparency in all directions
   - Prefer direct communication

- Create flow and rythm

IV. **Supportive leadership:**

- Set objectives and provide support
- Decentralize control structures
- Cultivate the change and change the culture

V. **Continuous Improvement:**

- Inspect and adapt the product
- Inspect and adapt the development process
- Inspect and adapt the organisation

These thirteen principles are then applied using the Agile Scaling Cycle (Figure 3.12).[67] As you can see in Figure 3.12, it is a three step cycle, starting with "reduce dependencies", where the dependencies between the development teams are reduced as much as possible. After that, the teams have to coordinate the remaining dependencies that could not be removed. During work in the second step, (organisational) problems might occur, which get removed in the third step, before the cycle continues again.

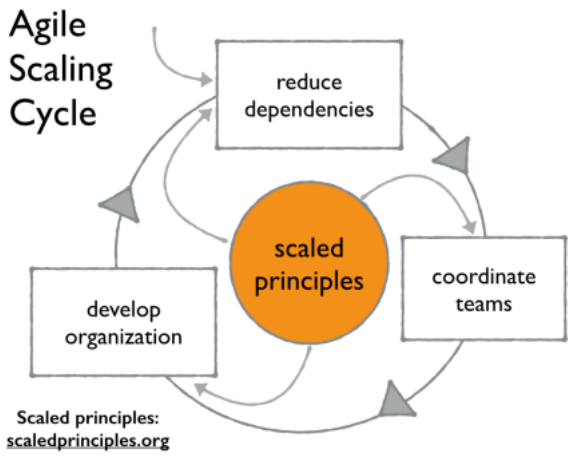


Figure 3.12.: The Agile Scaling Cycle[68]

3.20. Scrum at Scale (S@S)

Scrum at Scale is a minimal expansion on scrum, to scale it to the enterprise level. It focusses on keeping the modularity and the inherent object orientation of scrum. Scrum at Scale concentrates only on the pure scaling part of scaling agile practices. It does not give great detail on problems of distribution of teams and saturation of scrum in the company culture. Like scrum, the Scrum at Scale framework is modular.[66]
As seen in Figure 3.13, S@S consists of ten modules, linked in a Product Ownership and a Scrum Master cycle. Each module has defined goals, required input and expected outcomes, depending on its role. Each module is maintained at at least one level of the organization (Team, Business, Enterprise), where the responsibility for the module lies. The actual method to achieve the goals of a module are interchangeable, as long as they fulfill the defined goals with the predetermined input, leading to the desired output.[19]

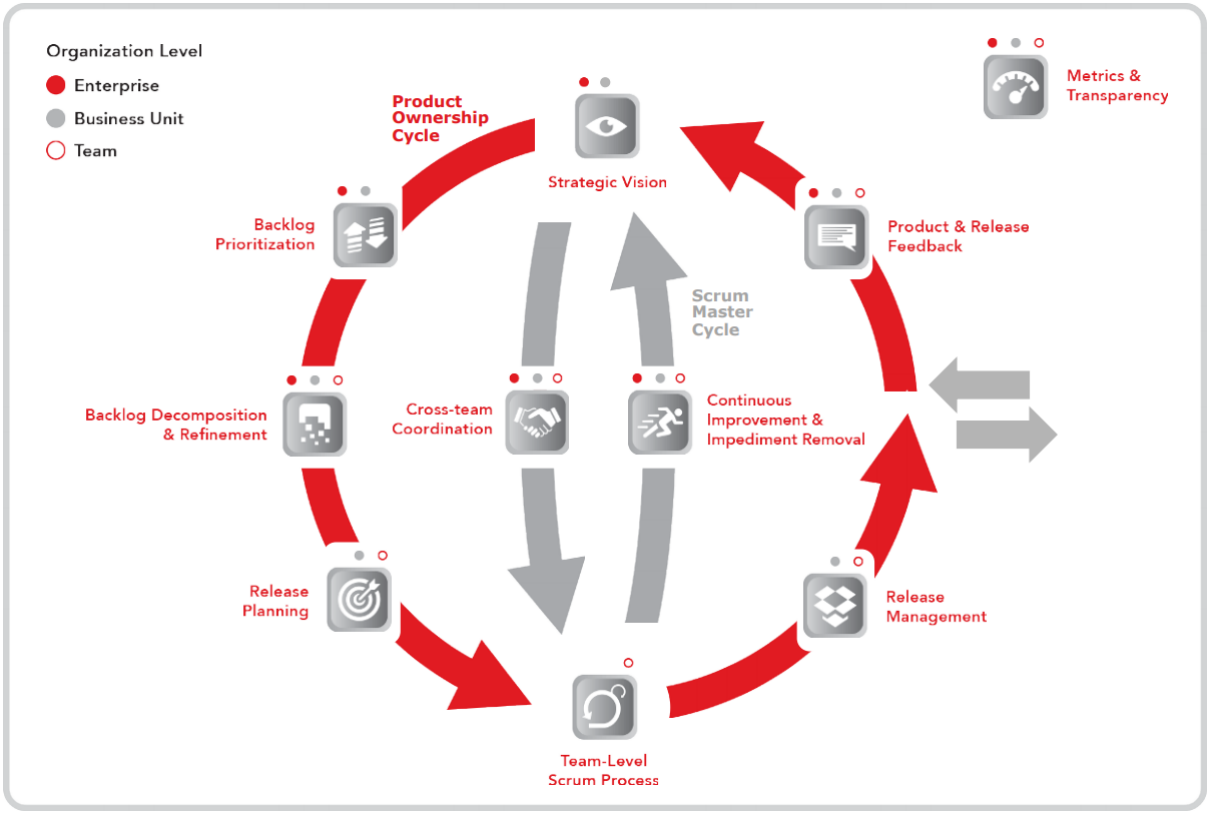


Figure 3.13.: The Scrum at Scale framework [19]

3.21. Scrum-of-Scrums (SoS)

Scrum of Scrums, also called Meta Scrum, is used to scale normal scrum by introducing, as the name suggests, a new scrum meeting: the Scrum of Scrums (also called Meat Scrum). [6]
As seen in Figure 3.14, SoS consists of each team sending an "ambassador" to a second (daily) scrum meeting, where the ambassadors of each scrum team, using a backlog of their own, discuss their teams' progress as well as trying to reduce overlap between the teams backlogs, as well as the coordination of the different scrum teams. This method can be scaled indefinitely, by creating new Scrums of Scrum of Scrums, where the teams of team-ambassadors choose new ambassadors, which will meet with their respective counterparts.[80]
These Meta-Scrums are ment to be a synchronization meeting between the teams, not a status reporting meeting for management though.[58]
The size of SoS meeting participants, as well as the meetings duration and frequency are not written in stone, but variable. In practice, the most common approach is a 30-60 minute SoS daily-meeting with 5-9 participants.[58]

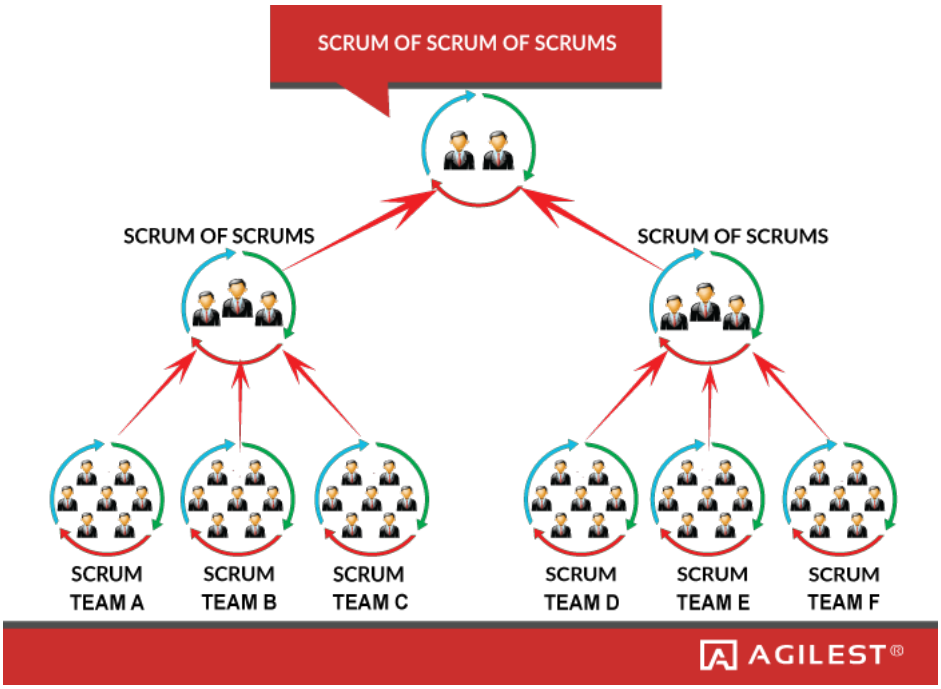


Figure 3.14.: The Scrum of Scrums structure[4]

3.22. Spotify Model (Spotify)

The Spotify Model is the method deployed at Spotify to manage multiple agile teams in different locations working on the same product. Its focus lies on the structuring and coordination of teams.[25]
As seen in Figure 3.15, a company using the Spotify Model is composed of squads. Those squads are comparable to scrum-teams, in that they are cross-functioning, self organizing teams with a product owner, working autonomously on a specific part of the product. These squads can however employ any method they see fit to develop their part of the product (like Scrum, Kanban, etc.).[9]
Squads working in a related area are grouped into Tribes. Each Tribe is co-located and not bigger than Dunbars Number (100 people) and is managed by a tribe lead, who is responsible for providing the best possible habitat for the squads within the tribe. To deal with dependencies between the squads and as a synchronization tool, ”on demand” Scrum of Scrums are held.[37]
To make use of economies of scale, chapters and guilds are formed, to profit from the experience of others in the same field and to foster communication between the teams. A chapter consists of all employees with similar skills working in the same general competency area, within the same tribe. Each chapter meets regularly to discuss specific challenges. The leader of each chapter takes on ”traditional manager responsibilities”, such as setting salaries and developing people, whilst still being involved in day-to-day work.[37]
Guilds span across tribes and can best be described as ”interest groups”. Guilds often include all chapters specific to the corresponding interest, like testing, agile coaching, etc. but are not limited to them, as anyone can join a guild. [37]

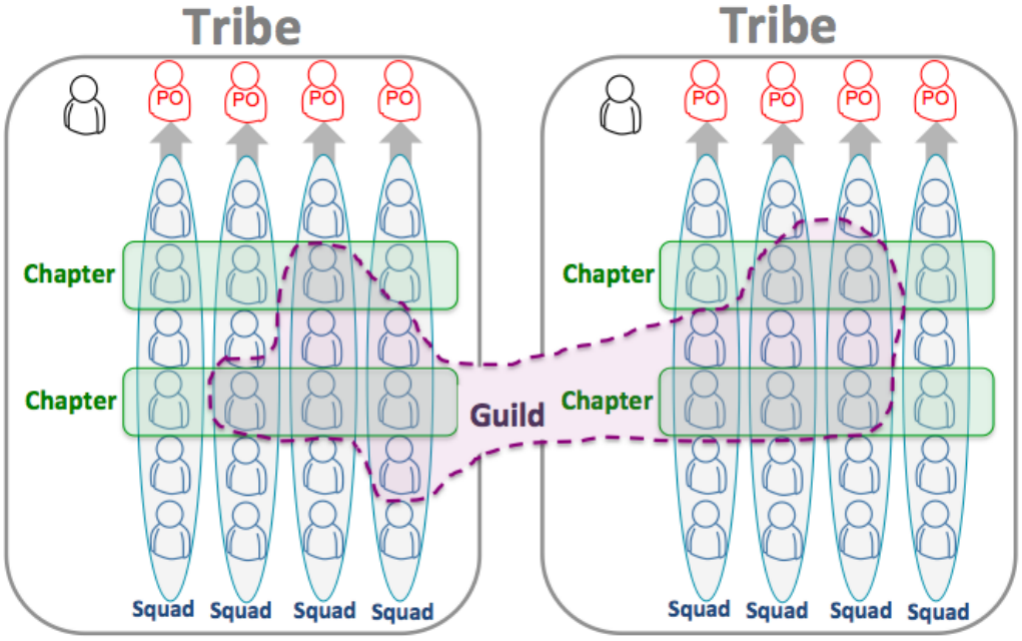


Figure 3.15.: The Spotify Model [37]

Part IV.

The Evolution of Scaling Agile Frameworks

4. The Evolution Of Scaling Agile Frameworks

The following chapter includes the expert interviews we held with the methodologists of the scaling agile practices we discovered in part III: Existing Frameworks. We start by presenting the questionnaire we used to conduct the interview online, as described in our research process.
The insights and information we gathered from this will then be presented. First in the descriptive statistics about the survey, then in the evaluation of the interview. Finally we will present an Evolution Map to visualize the chronological evolution of the identified practices, answering the second research question: "How did scaling agile practices develop?" and several Foundational Methods and Practices Maps, showcasing the relations between the practices and agile methods, visualizing the answer to the third research question:"On which foundational methods and practices are scaled agile practices based on?"

4.1. Questionnaire

As recommended by Punter et. al.[61], to create our survey we first defined the goals of the interview, which were:

I. Find out general information about the scaling agile practice from the view of its developers

II. Get an overview over all releases and chronological data about the practice, even the "unofficial" ones

III. Gather information about the agile and lean foundations of the practice

IV. Get the developers view on the combination of his practice with others

These four goals were then transformed into a set of 15 questions, some of which were further split up to ease the interview process and to gain more detailed information. Additionally some questions included follow up questions, depending on the answer of the developer, therefore resulting in a total of 22 questions asked via the unipark tool "questback" [31], with wich the interview was conducted. All 22 questions asked can be found in the appendix of this thesis (A.1).

4.2. Descriptive Statistics

As seen in Table 4.1, of the 21 invited main-developers of our identified scaling agile practices, 12 accepted the invitation and started the survey, resulting in a coverage rate of

Total number of developers	Coverage Rate	Completion Rate	Average time to complete	Duration
21	57.14%	23.81%	35 min.	58 days

Table 4.1.: Statistics of the survey

57.14%. Of those 12 participants 5 completed the whole survey, resulting in a completion rate of 23.81%.
The survey took the participants around 35 minutes to complete on average (arithmetic mean) and was held from the 24th of august 2017 to the 20th october 2017, resulting in a survey duration of 58 days, or roughly two months.
On the 1st of october 2017, a reminder was send to all developers that did not answer the survey, to increase the coverage rate. The developers of the following five Scaling Agile Practices completed the survey and are the focus of the following sections: Holistic Software Development (HSD), the Mega Framework (Mega), LargeScaleScrum (LeSS), FAST agile (FAST) and the Enterprise Transition Framework (ETF).

4.3. Evaluation

The answers to questions XIII to XVII of the survey can be found in tables 4.2 to 4.6, providing us with the insight into the foundational methods and practices of the Scaling Agile Practices. The data from these tables was used to generate the corresponding foundational methods and practices maps (Fig. 4.2 to 4.6), which are described in part 4.5.
In this section we will analyse the more openly asked questions of the survey and provide an insight into the minds of the main developers of ETF, LeSS, HSD, FAST and Mega:

Agile Activities					
	ETF	FAST	HSD	LeSS	Mega
Agile Portfolio Planning	X	-	X	X	X
Daily Standup	X	-	X	X	X
Iteration Planning	X	X	X	X	X
Iteration Reviews	X	X	X	X	X
Product Roadmapping	X	-	X	X	X
Release Planning	X	-	X	X	X
Retrospectives	X	-	X	X	X
Story Mapping	X	X	X	X	-
Team-based Estimation	X	-	X	X	X

Table 4.2.: Agile Activities

ETF: The Enterprise Transition Framework, as described by its main developer, provides a way to design organizations. These can be organizations of any industry, although more traditional sectors where all the work can be predetermined and executed without any mistakes, may be better off with automation. Following this, ETF challenges the popular belief that organizations are machines producing or delivering a product, but proposes that they are more networks of competing interests. ETF therefore focusses on a natural systemic view.
The Enterprise Transition Framework employs safe-to-fail projects to foster complexity awareness and discovery, 6 principles to guide change and organizational design and a library of successfully used patterns to scale agile.
The aim here is to change the culture of the organization and to align culture and strategy, but it also tries to keep the culture heterogenised enough to foster innovation.
The major challanges, problems and misunderstandings that may erupt during the application of ETF are listed as: a lack of mindset change, the fall back to old plan and execute approaches and not allowing mistakes to happen and not to create safe-to-fail environments. It is also mentioned that it requires the understanding of Lean, Agile and Cynefin (a sense making framework for managing complexity), which takes a lot of time.
In summary, ETF is a Set of Principles designed to change the culture of an organization into a self-improving and experimenting one that is aligned with the strategy of the organization, independent of a specific practice to develop software, which can be combined with different other Scaling Agile Practices (Figure 4.6).

Agile Artifacts					
	ETF	FAST	HSD	LeSS	Mega
Backlog Items	X	-	X	X	X
Information Radiators	X	X	X	X	X
Kanban Board	X	-	-	-	X
Product Backlog	X	-	X	X	X
Product Increment	X	-	-	X	X
Product Roadmap	X	-	X	X	X
Product Vision Statement	X	-	X	X	-
Release Plan	X	-	X	X	X
Sprint Backlog	X	-	X	X	X
User Stories	X	-	X	X	X

Table 4.3.: Agile Artifacts

LeSS: As described by its developer, the idea behind LeSS is creating an organizational design, strongly based on Scrum & Lean Thinking, that for the case of many teams working together on one product, is consistent with highest adaptiveness/agility and working on highest value. All of this in the context of not knowing for sure what is highest value, a changing environment which can lead to changing highest value and needing an empirical system for product and processes.
The goals are to achieve low cost of change and low transaction cost for a delivery cycle through high adaptiveness/agility and to be working on highest value.
To achieve this, LeSS changes the organization and removes traditional elements like portfolios and programs and utilizes Scrum as the main development method.
Major challenges in adopting LeSS are considered to be following Larman's five Laws:[39]

I. Organizations are implicitly optimized to avoid changing the status quo middle- and first-level manager and "specialist" positions and power structures.

II. As a corollary to (I), any change initiative will be reduced to redefining or overloading the new terminology to mean basically the same as status quo.

III. As a corollary to (I), any change initiative will be derided as "purist", "theoretical", "revolutionary", "religion", and "needing pragmatic customization for local concerns" - which deflects from addressing weaknesses and manager/specialist status quo.

IV. As a corollary to (I), if after changing the change some managers and single-specialists are still displaced, they become "coaches/trainers" for the change, frequently reinforcing (II) and (III).

V. Culture follows structure.

As far as incompatibility is concered, LeSS lists local optimization, single-function teams/roles, projects and programs, project and program management and annual operating budgets driving annual release plans as the main incompatibilities.
In summary, LeSS provides a lightweight organizational design to change the software development process as well as the whole organization, with a strong focus on systems thinking.

Agile Methods					
	ETF	FAST	HSD	LeSS	Mega
Agile Software Development	X	X	X	X	X
Adaptive Software Development	-	-	-	-	-
Agile Modeling	-	-	X	-	-
Agile Unified Process	-	-	-	-	-
Disciplined Agile Delivery	-	-	-	-	-
Dynamic Systems Development Method	-	-	-	-	-
Extreme Programming	X	-	-	X	-
Feature-Driven Development	-	-	-	-	-
Iterative Development	X	-	X	-	-
Kanban	-	-	-	-	-
Lean Software Development	X	-	X	X	-
Rapid Application Development	-	-	-	-	-
Scrum	X	-	X	X	X
Scrumban	-	-	-	-	-

Table 4.4.: Agile Methods

HSD: The idea behind HSD is stated as:"Combining the philosophies of agile and lean with organisational workflow to achieve whole organisation agility. HSD bridges the traditional business management world with software agility in teams, without compromising the goals of either." As seen in Figure 4.2, HSD, like ETF, can be combined with other Scaling Agile Practices, as it does not specify a specific method of software development. Its focus is on cross-team integration, delivering value and embracing change throughout the whole business. HSD has significant architectural depth, psychology-based change practices and is less about planning than other models. It covers everything from business strategy to version-control.
HSD presents improved flow throughout the business, making work healthier, faster, cheaper and happier as its main goals.
Big issues in adoption are trying to have the organization adopt all of HSD, as it is more a library of guidance rather than a method to adopt and that its comprehensiveness and that everything in it is linked together.
To provide a summary, HSD provides organization-wide positive, people-centric change wich can be combined with other Scaling Agile Practices.

Agile Practices					
	ETF	FAST	HSD	LeSS	Mega
Acceptance Test-Driven Development	X	-	X	X	-
Agile Modeling	-	-	X	X	-
Agile Portfolio Planning	X	-	X	X	-
Agile Testing	X	-	X	X	X
Agile/Lean UX	X	-	X	X	X
Behavior-Driven Development	X	-	-	X	X
Coding Standards	X	X	-	X	X
Continuous Deployment	X	X	X	X	X
Continuous Integration	X	X	X	X	X
Cross-Functional Team	X	X	X	X	X
Dedicated Product Owner	X	X	X	X	X
Domain-Driven Design	-	-	-	X	X
Iterative and Incremental Development	X	X	X	X	X
Open Work Area	X	X	-	X	-
Pair Programming	X	X	-	X	-
Planning Poker	X	-	-	X	X
Product Roadmapping	X	-	X	X	-
Refactoring	X	X	-	X	X
Single Team (integrated dev & testing)	X	-	X	X	-
Story-Driven Modeling	-	-	X	X	-
Test-Driven Development	X	X	X	X	X
Timeboxing	X	-	-	X	X
Unit Testing	X	X	X	X	X
User Story Mapping	X	X	-	X	-
Velocity Tracking	-	-	-	X	X

Table 4.5.: Agile Practices

FAST: "The idea behind FAST Agile is to move teams that need to cooperate into a network. This network becomes a complex adaptive system and self-organizes people around work by means of Open Space Technology." FAST focusses on distributing work across multiple teams and keeping them in synch with each other in an agile way, rather than using hierarchical centralized mechanisms commonly used in command and control structures.
It does so to deliver rapid releases of production code, short feedback cycles, empowered and highly motivated employees, cross training of skills, improve natural leadership, innovation, happiness and to lower the headcount.
As seen in Table 4.7, FAST scales only to the Program and Team level, providing a focus on delivering software in an agile manner, rather than changing the whole organization.
This is also reflected in the challenges that arise in adopting FAST: FAST has no scrum master role, so it's often rejected where Scrum is strongly embraced and entrenched. Also, FAST creates a vacuum where work happens through self-organization, leading to a challenge for people that are used to working in an environment of command and control.

And maybe the biggest benefits as well as problems arise from the fact that FAST requires co-location.
In summary, FAST is a mechanism organizing people in self-organizing networks, using Open Space Technology, to increase agility in the software developing process, and can be used as a template by other Scaling Agile Practices like ETF and eScrum.

Agile Principles					
	ETF	FAST	HSD	LeSS	Mega
Build Quality In	X	X	X	X	X
Continuous Attention to Technical Excellence	X	X	X	X	X
Create Knowledge	X	X	X	X	X
Defer Commitment	X	X	X	X	-
Deliver Fast	X	X	X	X	X
Eliminate Waste	X	X	X	X	-
Face-to-Face Conversation	X	X	X	X	X
Frequent Releases	X	X	X	X	X
Optimize The Whole	X	X	X	X	-
Regular Self-Reflection Builds the Best Teams	X	X	X	X	X
Respect People	X	X	X	X	X
Statisfy the Customer Though Early Delivery	X	X	X	X	-
Self-Organization Leads to the Best Outcome	X	X	X	X	X
Short Iterations	X	X	X	X	X
Simplicity	X	X	X	X	X
Trust and Motivate Employees	X	X	X	X	-
Welcome Change Requirements	X	X	X	X	-
Working Software as Primary Measure of Success	X	X	X	-	X

Table 4.6.: Agile Principles

Mega: The main goal of the Mega framework was stated to be to keep information flowing among teams, to anticipate issues, leverage synergies and help each other. It can accommodate many different agile processes inside and is highly adaptable, resembling in some ways the Spotify Model. In Mega, teams deploy their changes independently, while focussing on the essence of Mega: communication.
Benefits of using Mega are listed as: higher levels of communication and overall alignment, less incompatibilities, less integration errors, less defects and less rework, more perceived value of product changes and faster spotting of lesser value changes and higher team cohesion.
Mega scales up to the Portfolio level (Table 4.7), providing a framework that scales with the underlying architecture and that applies to teams and their activities.
The Mega Framework can be combined with most lightweight practices, since the Mega Framework just goes into more detail on how the ceremonies should happen and their purpose. As Scrum-of-Scrums was the inspiration for Mega Framework, combining the two is seen to be a good idea by the main developer. The software development process in Mega is also variable, as often Kanban is used, but Scrum and others can be used as well.
Overall Mega delivers tools to better define the purpose and procedure of ceremonies from the team to the portfolio level and can be used in such a manner by frameworks like ETF and HSD.

Scaling					
	ETF	FAST	HSD	LeSS	Mega
Team	X	X	X	-	-
Program	X	X	X	-	-
Portfolio	X	-	X	-	X
IT Organization	X	-	X	-	-
Enterprise	X	-	X	X	-

Table 4.7.: The Scale of Scaling Agile Practices

In this section we gave an insight into the mind of the methodologists of the Enterprise Transition Framework, LeSS, Holistic Software Development,FAST Agile and the Mega Framework, showing their thoughts behin their Scaling Agile Practices. We showed, and will show in greater detail in section 4.5, that the five Scaling Agile Practices we gathered data from have a lot in common. While the goal of all five Scaling Agile Practices is to scale the agile principles to big organizations, the way they achieve this is unique to each and every one of them. While some of them change the whole organization from top to bottom and leave no stone standing, others fit into a specific part of the organization and perform their change there. Despite their commonalities, there are big differences in scale, applicability with other frameworks, the software development process, and the overall focus of the Scaling Agile Practice. Although the way they achieve their goal is different, the problems they face are mostly the same: an overall change resistance, best described by the five laws of larman[39].

4.4. Evolution Map

In this section we present the Evolution Map, answering our second research question "How did scaling agile practices develop?". We take a look at the starting points of each Scaling Agile Practice, their development over the years and where they currently are, as well as provide an overview over Large-Scale Agile in the whole. Based on the work of Uludag et al.[54] and the results of our survey, we visualized all the chronological information about the identified scaling agile practices in the Evolution Map (Figure 4.1), showcasing the start of development, the different verions since then, the current version and the influences that drove the development of certain versions.
From this we can clearly see that the roots of Large-Scale Agile Development lie way back in the 1990s with the development of the Crystal Methods and DSDM. After the publication of the Agile Manifesto in 2001 [24], we can see an increase in the early and mid 2000s. With Agile Practices as a whole picking up steam and getting more and more traction in the software development community over the years, we can clearly make out a boom of Scaling Agile Practices in and around 2014. We can determine that of all practices that took part in our survey, LeSS is the oldest, having been released in 2005. As stated in our survey, LeSS will always be only one release. "As with scrum, the focus is not in changing large-scale scrum (which is simple), but in changing the organization." HSD and Mega on the other hand follow a different approach, as they have many releases, being in a state of constant self-improvement, adaption and change.

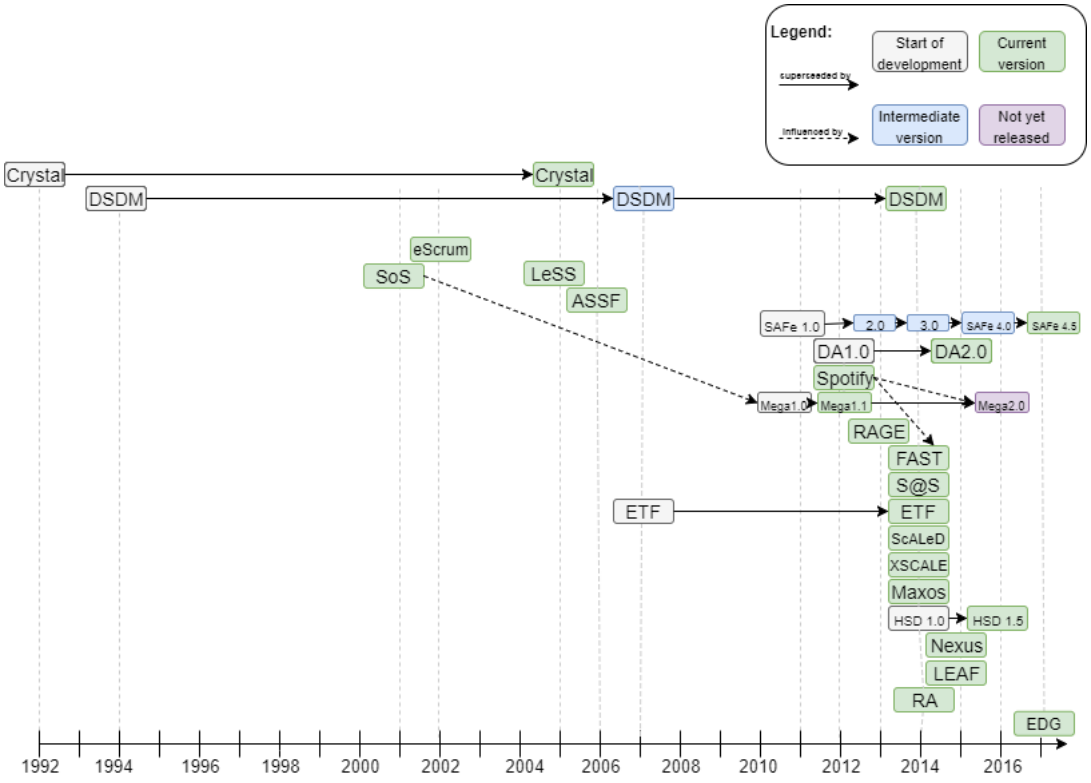


Figure 4.1.: The Evolution Map

4.5. Foundational Methods and Practices Maps

Based on the information gathered in the survey, we set out to illustrate the differences and dependencies in the Scaling Agile Practices we got information from. In the following Circos-Tables [38], we show the foundational Methods, Practices, Principles, Artifacts and Activities that these Scaling Agile Practices are based on. The direction of the relationships go from the Scaling Agile Practice to the base of the Methods/Practices/Principles/Artifacts or other Scaling Agile Practices that they have a relation with.

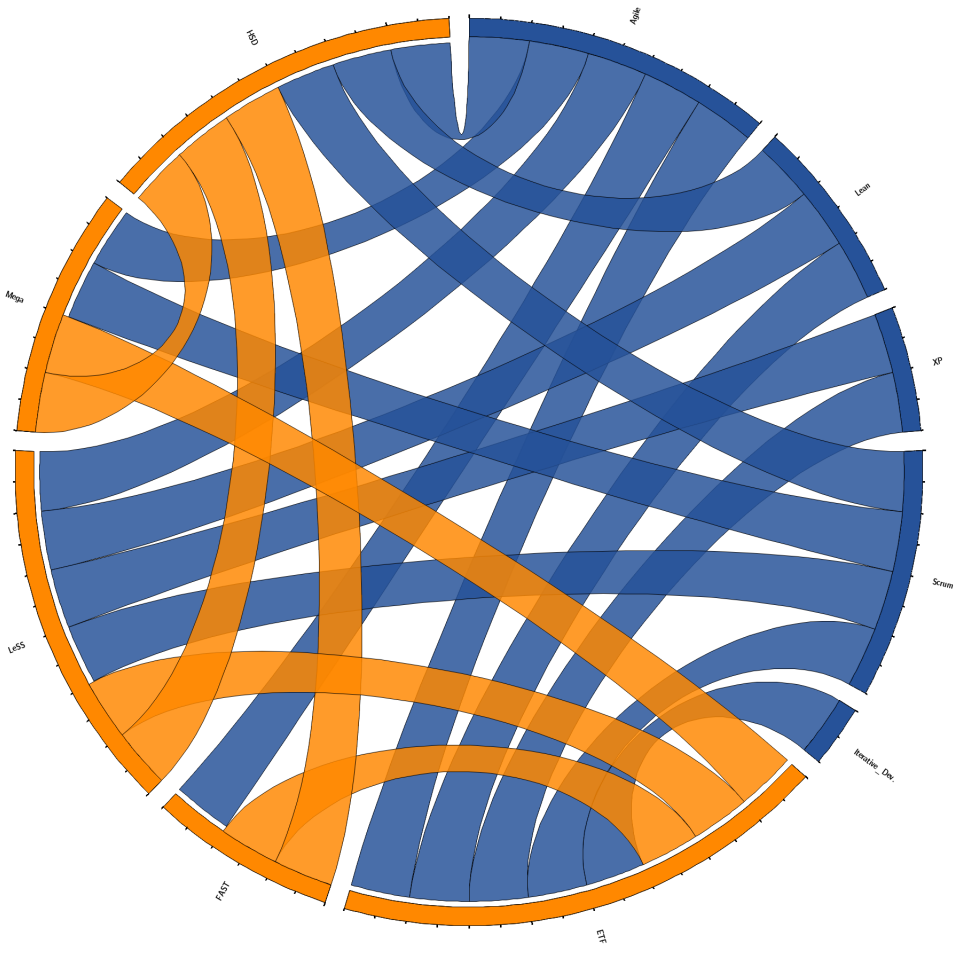


Figure 4.2.: The Methods Map

4.5.1. Methods

Figure 4.2 shows the agile methods the five Scaling Agile Practices in our survey are based on and, as all following Figures, the relationship between the Scaling Agile Practices. In

this Figure we can see that HSD and ETF can be combined with Mega, LeSS or FAST, to adapt them to a Large Scale organization. We can see that ETF is the only practice using Iterative Development, while the only method FAST is based on is Agile, which is the foundation of all five Scaling Agile Practices.

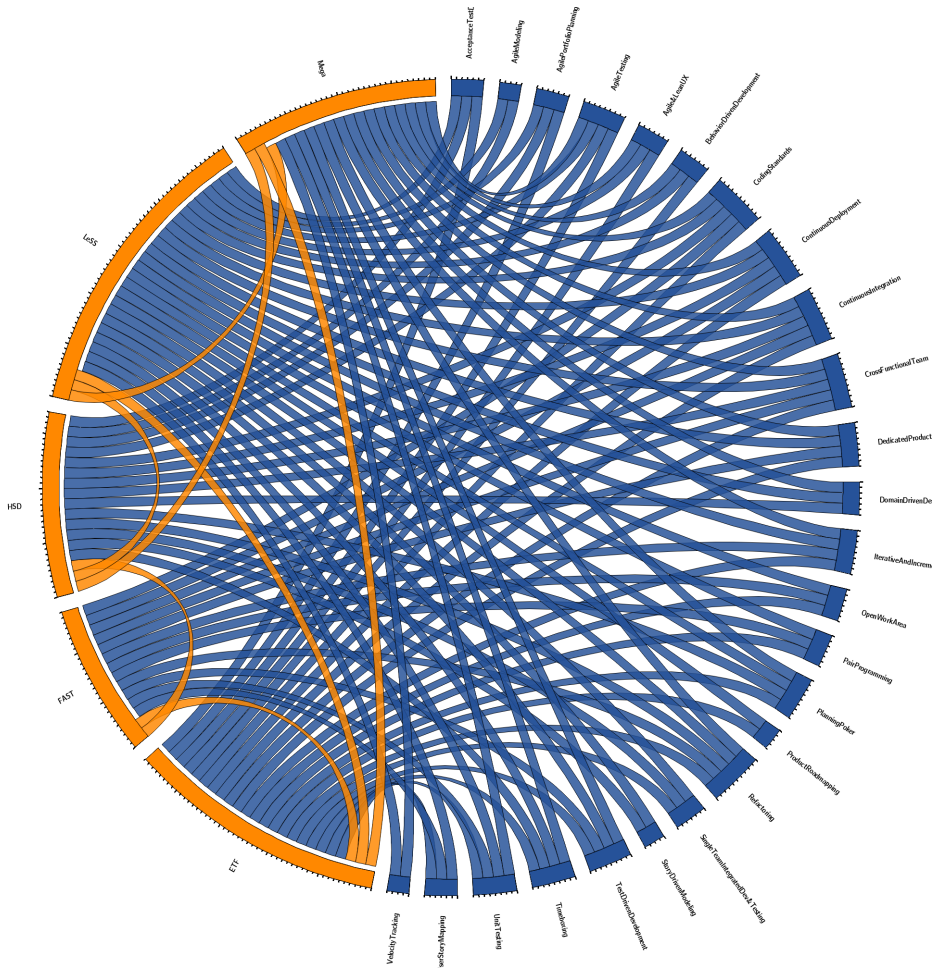


Figure 4.3.: The Practices Map

4.5.2. Practices

Being in stark contrast to the smaller Methods Map (Fig. 4.2) in terms of size, the Practices Map 4.3 shows that FAST is based on the least practices, while Velocity Tracking is the least employed practice. Here we can again see the relations between the Scaled Agile Practices with Mega being able to make use of the LeSS framework.

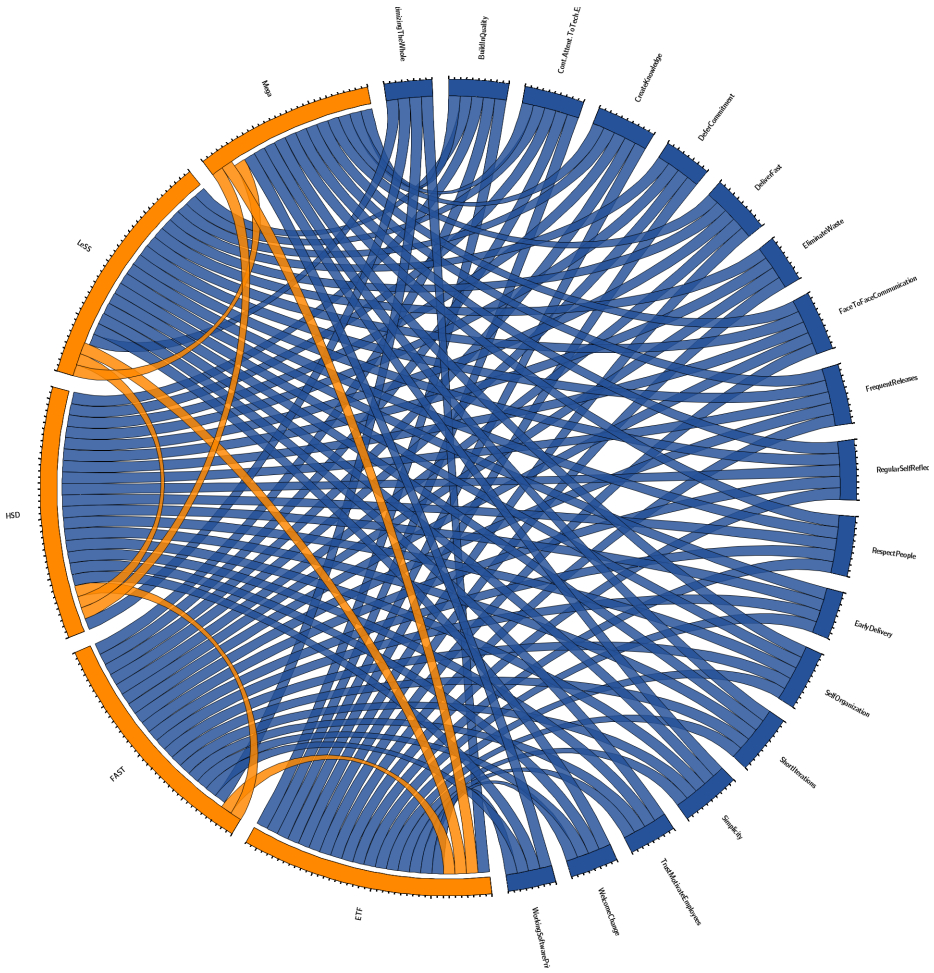


Figure 4.4.: The Principles Map

4.5.3. Principles

While the two afore mentioned maps show clear differences in employed methods and practices, the Principles Map (Figure 4.4) shows that all five Scaling Agile Practices have roughly the same amount of foundational principles, while no principle is the basis of less than four Scaling Agile Practices.

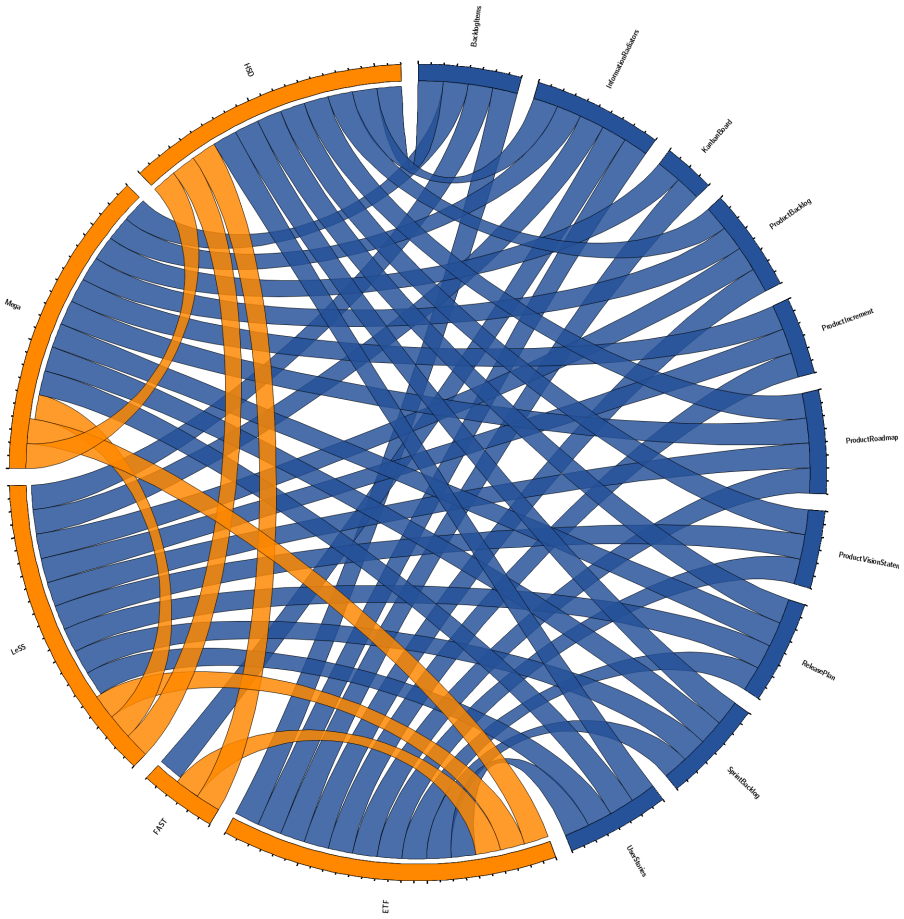


Figure 4.5.: The Artifacts Map

4.5.4. Artifacts

Displaying the different artifacts used in our five Scaling Agile Practices, the Artifacts Map (Figure 4.5) shows that the only artifact used in FAST are Information Radiators, while ETF uses all ten artifacts identified. One can also see that Kanban Boards are only employed in ETF and the Mega framework.

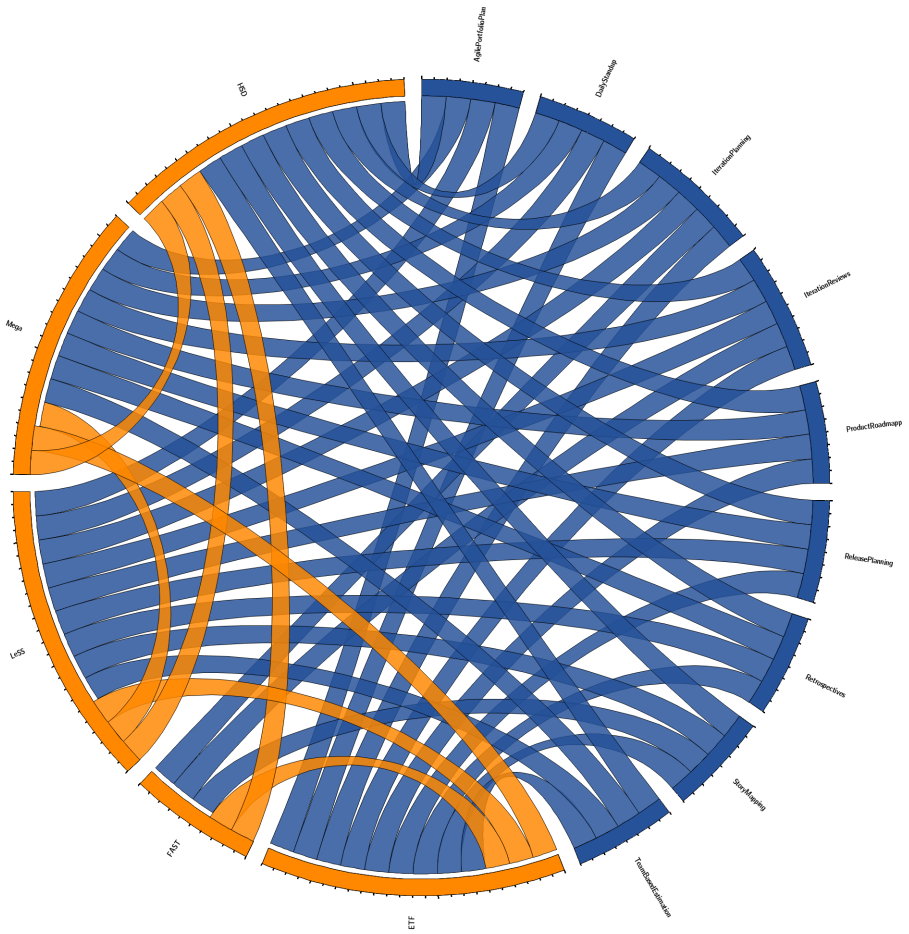


Figure 4.6.: The Activities Map

4.5.5. Activities

The Activities Map (Figure 4.6) at last shows a fairly even distribution of agile activities, where no activity is used by less than four Scaling Agile Practices. It also reveals that, while all other Scaling Agile Practices employ almost all of the agile activities, FAST only employs Iteration Planning, Iteration Reviews and Story Mapping, while leaving out Team Based Estimation, Retrospectives, Release Planning, Product Roadmapping, Daily Standups and Agile Portfolio Planning.

Part V.

Discussion

5. Discussion

5.1. Key Findings

We identified the following 22 different Scaling Agile Practices in October of 2017:

I. Agile Software Solution Framework (ASSF)

II. Crystal Family (Crystal)

III. Disciplined Agile 2.0 (DA 2.0)

IV. Dynamic Systems Development Method Agile Project Framework for Scrum (DSDM)

V. Enterprise Scrum (eScrum)

VI. Enterprise Transition Framework (ETF)

VII. Event Driven Governance (EDG)

VIII. eXponential Simple Continuous Autonomous Learning Ecosystem (XSCALE)

IX. FAST Agile

X. Holistic Software Development (HSD)

XI. Large Scale Scrum (LeSS)

XII. Lean Enterprise Agile Framework (LEAF)

XIII. Matrix of Services (Maxos)

XIV. Mega Framework (Mega)

XV. Nexus

XVI. Recipes for Agile Governance in the Enterprise (RAGE)

XVII. Resilient Agile (RA)

XVIII. Scaled Agile Framework 4.5 (SAFe 4.5)

XIX. ScALeD Agile Lean Development (ScALeD)

XX. Scrum at Scale (S@S)

XXI. Scrum-of-Scrums (SoS)

XXII. Spotify Model (Spotify)

Through the survey and the literature review we conducted, we were able to locate all 22 Scaling Agile Practices and their evolution on a Time Map (Figure 4.1).
Further we were able to construct an Activities (Figure 4.6), Practices (Figure 4.3), Artifacts (Figure 4.5), Methods (Figure 4.2) and Principles Map (Figure 4.4) to showcase the foundational methods and practices of the Enterprise Transition Framework, Holistic Software Development, FAST Agile, LeSS and the Mega Framework.
We gathered an insight into the minds of the methodologists of those Scaling Agile Practices and the thought process behind them. We discovered that, while most of them share a lot of commonalities, each adresses a certain problem a certain way, that is unique to each Scaling Agile Practice. We also could pinpoint which Scaling Agile Practices can be used by which one, expanding them or serving as a template for the software development process.

5.2. Limitations

In this section we highlight the limitations of this work and its applicability. First, there is the obvious limitation that this thesis only provides a snapshot of the Scaling Agile Practices "marketplace" in 2017. In the future there may be new Scaling Agile Practices and old ones might no longer be supported or be used.
Secondly, although we conducted the literature review with utmost rigour, we might not have identified all Scaling Agile Practices, as they might have fallen through our defined search terms, have been published in databases or libraries where we did not have access to, or might be published in languages other than english and german and were therefore out of our reach.
To gain more access to the ideas of the developers of the Scaling Agile Practices and transport our goals accross to the interviewee, an expert interview in the form of an actual interview via telephone or in person might have been better than an online survey.
In the survey, we were limited in the amount of interviewees to one per Scaling Agile Practice, reducing the validity and chance of response. This lead to a rather low sample size of 5 Scaling Agile Practices out of 22, whose data would then be used. Due to time constraints, we could only send out one reminder, instead of the recommended three, and were only able to hold the survey over 58 days from the months of august to october.
Another reason for the low completion rate of 23.81% might have been the high amount of questions that were required to be answered, as well as the rather long duration of around 35 minutes on average to complete the survey.
The therefore following description of the Scaling Agile Practices and their foundations was based mainly on the answers given by the developers. Not in the scope of this thesis was the comparison or the categorization into groups according to different criteria.
With the Evolution Map only showcasing years, a lot of information was lost in favor of readability and clarity.
Lastly, a more detailed answer of the second , as well as the answer to the third research question was only based on the answers provided by the methodologists in the survey, resulting in research question three only focussing on five Scaling Agile Practices instead of all 22 identified Practices.

Part VI.

Conclusion

6. Conclusion

6.1. Summary

In this thesis we first defined its scope by asking the following three research questions:

I. Which scaling agile practices exist?

II. How did scaling agile practices develop?

III. On which foundational methods and practices are scaled agile practices based on?

These research questions were then answered by conducting an extensive literature review in five steps, as recommended by Brocke et al. [86]. Through this we identified 22 different Scaling Agile Practices in 2017.
After the literature review was conducted, an expert interview of the form of an online survey of the main methodologists was conducted. This survey was held over the course of two months, following the principles of Punter et al. [61]. It resulted in a 23.81% completion rate, with the completing Scaling Agile Practices being the Enterprise Transition Framework, LeSS, Holistic Software Development, FAST Agile and the Mega Framework. These five were then used to answer research question II and III. Following the data gathered by the literature review and the survey, a Time Map showcasing the Evolution of all Scaling Agile Practices was constructed.
Focussing on the five Scaling Agile Practices ETF, LeSS, HSD, FAST and Mega, we showcased the idea behind these Scaling Agile Practices according to their developers. We showed the main challenges these Practices face in the eyes of their creators, as well as their relationships between each other, as well as to the methods, artifacts, activities, principles and practices they are based on. This information was then visualized in five Foundational Maps.

6.2. Outlook

We hope that this work will serve as a guide to Scaling Agile Practices, providing an overview over what is out there and over the matureness of these Practices, as well as their foundations. We realise that in order for this to be the case, the answers to the research questions asked in this thesis, especially the first, need to be updated in the future, as this work is only a snapshot of 2017. In regular intervals a literature review using the same process could be used to keep the list of Scaling Agile Practices up to date.
An improved version of the survey could be established with an ongoing duration, to gain insight into each new Scaling Agile Practice. In order to get this information from existing Scaling Agile Practices that did not take part in our survey, a different approach in doing so could be used.
Another way to build on this thesis would be to get the view of practitioners of the Scaling Agile Practices that were identified in this work, and compare them with the expectations of the methodologists. One could also provide a comparison of the identified Scaling Agile Practices expanding on the similarities and differences we could identify, to provide a guide for companies to choose a Practice that fits their needs. This comparison could also be used to group the frameworks according to empirically chosen criteria.

Part VII.

Appendix

A. Appendix

A.1. Questionnaire questions

The following are all 22 questions that were used in the questionnaire of scaling agile practice developers:

I. Which scaling agile practice did you develop?

II. What is the idea behind your scaling agile practice and what problems does it mainly address?

III. How would you categorize your scaling agile practice?

IV. What outcomes are expected after using your scaling agile practice?

V. On which level does your scaling agile practice scale?

VI. What are the major challenges, problems, and misunderstandings in implementing and using your practice?

VII. When did you start with the creation of your scaling agile practice?

VIII. When did you publish or release the first version of your scaling agile practice?

IX. How many releases does your scaling agile practice have and when were they released?

X. Do you plan to continue developing your scaling agile practice?

XI. Why do you want to stop the further development of your scaling agile practice?

XII. What is your advice for people interested in your scaling agile practice, which is not developed anymore? Should they continue to use it or should they use other scaling agile practices?

XIII. What agile activities does your scaling agile practice use?

XIV. What agile methods does your scaling agile practice use?

XV. What agile practices does your scaling agile practice use?

XVI. What agile artifacts are used in your scaling agile practice?

XVII. What principles does your scaling agile practice employ?

XVIII. What differentiates your scaling agile practice from others?

XIX. With which other scaling agile practices can your scaling agile practice be combined? Are there major problems involved in combining? Are there major gains for combining?

XX. What are the main changes between your scaling agile practice versions and were they influenced by other scaling agile practices?

XXI. Is there anything your scaling agile practice is incompatible with?

XXII. Do you have further remarks or some points to add?

Bibliography

[1] DasScrumTeam AG. http://scaledprinciples.org, October 2017.

[2] DasScrumTeam AG. https://www.dasscrumteam.com/scaled, October 2017.

[3] agile42. http://www.agile42.com/en/agile-transition/etf/, October 2017.

[4] Agilest. https://www.agilest.org/scaled-agile/scrum-of-scrums/, October 2017.

[5] Muhammad Ovais Ahmad, Jouni Markkula, and Markku Oivo. Kanban in software development: A systematic literature review. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pages 9–16. IEEE, 2013.

[6] Agile Alliance. https://www.agilealliance.org/, October 2017.

[7] 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.

[8] 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.

[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] Scott Ambler. Dr.dobb's journal's 2008 project success survey. http://www.ambysoft.com/surveys/success2008.html, 2008.

[11] Scott W. Ambler. The agile scaling model (asm): Adapting agile methods for complex environments. *Adapting Agile Methods for Complex Environments*, 2009.

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

[13] Kent Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999.

[14] Kent Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[15] Mike Beedle. Enterprise scrum-definition: Agile management for the 21 st century. *Enterprise Scrum Inc.*, 2015.

[16] Barry Boehm. A survey of agile development methodologies. *Laurie Williams*, 2007.

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

[18] 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.

[19] Alex Brown and Jeff Sutherland. Scrum at scale-go modular for greater success. In *Agile 2014 - Orlando*, 2014.

[20] The LeSS Company B.V. https://less.works/less/framework/index.html, October 2017.

[21] Alistair Cockburn. *Surviving object-oriented projects: a manager's guide*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1998.

[22] Alistair Cockburn. *Crystal clear: a human-powered methodology for small teams*. Pearson Education, 2004.

[23] cPrime. https://www.cprime.com/rage/, October 2017.

[24] Ward Cunningham. http://agilemanifesto.org, September 2017.

[25] Patrick Daut. agile@ scale: Do more with less or be safe? ansätze zur skalierung-ein überblick. In *Vorgehensmodelle*, pages 159–167, 2015.

[26] 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.

[27] Torgeir Dingsøyr, Tor Erlend Fægri, and Juha Itkonen. What is large in large-scale? a taxonomy of scale for agile software development. In *International Conference on Product-Focused Software Process Improvement*, pages 273–276. Springer, 2014.

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

[29] AQ Gill and Brian Henderson-Sellers. Measuring agility and adaptibility of agile methods: A 4 dimensional analytical tool. In *The IADIS international conference on applied computing 2006*. IADIS Press, 2006.

[30] AQ Gill, Brian Henderson-Sellers, and TM McBride. Agile adoption and improvement model. In *European, Mediterranean and Middle Eastern Conference on Information Systems*. Polytechnic University of Valencia, 2007.

[31] Questback GmbH. www.unipark.de, September 2017.

[32] Daniel R Greening. Enterprise scrum: Scaling scrum to the executive level. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–10. IEEE, 2010.

[33] Inc. ICONIX Software Engineering. http://www.iconixsw.com/a-better-agile-methodology.html, October 2017.

[34] Scaled Agile Inc. http://www.scaledagileframework.com/, October 2017.

[35] Scaled Agile Inc. http://www.scaledagileframework.com/portfolio-level/, October 2017.

[36] SCRUM ALLIANCE Inc. https://www.scrumalliance.org/why-scrum/scrum-guide, November 2017.

[37] Henrik Kniberg and Anders Ivarsson. Scaling agile@ spotify with tribes, squads, chapters & guilds. *Entry posted November*, 12, 2012.

[38] Martin Krzywinski. http://circos.ca, November 2017.

[39] Craig Larman. http://larmanslaws.org, November 2017.

[40] Craig Larman and Bas Vodde. *Large-Scale Scrum: More with LeSS*. Addison-Wesley Professional, 1st edition, 2016.

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

[42] Agile Business Consortium Limited. https://www.agilebusiness.org/content/principles, October 2017.

[43] Mark Lines and Scott Ambler. http://www.disciplinedagiledelivery.com/process/, October 2017.

[44] Leanpitch Technologies PVT LTD. https://leanpitch.com/lean-enterprise-agile-framework-leaf/, October 2017.

[45] Mike MacDonagh and Steve Handy. http://www.holistic-software.com/h-model, October 2017.

[46] Mike MacDonagh and Steve Handy. http://www.holistic-software.com/introduction/introduction-2, October 2017.

[47] Mike MacDonagh and Steve Handy. http://www.holistic-software.com/maps/big-picture, October 2017.

[48] Rafael Maranzato, Marden Neubert, and Paula Herculano. Scaling scrum step by step:"the mega framework". *agilealliance.org*, 2016.

[49] 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.

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

[51] Eric A. Marks. *A Lean Enterprise Governance Manifesto: Improving Business Performance with Event-Driven Governance*. Agile Path Corporation, 2012.

[52] Christoph Mathis. *SAFe-Das Scaled Agile Framework: Lean und Agile in großen Unternehmen skalieren*. dpunkt. verlag, 2016.

[53] Peter Merel. http://www.xscalealliance.org/index.html#manifesto, October 2017.

[54] Uludağ Ö., Kleehaus M., X. Xu, and Matthes F. Investigating the role of architects in scaled agile frameworks. In *21th Conference on Enterprise Distributed Object Computing (EDOC)*, 2017.

[55] Eric Overby, Anandhi Bharadwaj, and V Sambamurthy. Enterprise agility and the enabling role of information technology. *European Journal of Information Systems*, 15, April 2006.

[56] Harrison Owen. *Open space technology: A user's guide*. Berrett-Koehler Publishers, 2008.

[57] 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.

[58] Maria Paasivaara, Casper Lassenius, and Ville T Heikkilä. Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work? In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 235–238. ACM, 2012.

[59] Mary Poppendieck. Lean software development. In *Companion to the proceedings of the 29th International Conference on Software Engineering*, pages 165–166. IEEE Computer Society, 2007.

[60] Mary Poppendieck and Michael A Cusumano. Lean software development: A tutorial. *IEEE software*, 29(5):26–32, 2012.

[61] Teade Punter, Marcus Ciolkowski, Bernd Freimut, and Isabel John. Conducting online surveys in software engineering. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE'03)*, 2003.

[62] Ron Quartel. http://www.fast-agile.com, October 2017.

[63] Asif Qumer and Brian Henderson-Sellers. Construction of an agile software product-enhancement process by using an agile software solution framework (assf) and situational method engineering. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 1, pages 539–542. IEEE, 2007.

[64] 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.

[65] Linda Rising and Norman S Janoff. The scrum software development process for small teams. *IEEE software*, 17(4):26–32, 2000.

[66] Patrick Roach. https://www.scruminc.com/scrum-scale-case-modularity/, October 2017.

[67] Stefan Roock. https://stefanroock.wordpress.com/2014/07/13/agile-scaling-cycle-from-scaled-principles-to-practices/, October 2017.

[68] Stefan Roock and Peter Beck. Beyond safe: Wirklich agile und lean sein mit den scaled prinzipien. In *scaled vortrag las zurich 2014*, 2014.

[69] Doug Rosenberg, Barry Boehm, Bo Wang, and Kan Qi. Rapid, evolutionary, reliable, scalable system and software development: the resilient agile process. In *Proceedings of the 2017 International Conference on Software and System Process*, pages 60–69. ACM, 2017.

[70] K Schwaber. Nexus guide. *Scrum. org, August*, 2015.

[71] Ken Schwaber. Scrum development process. In *Business object design and implementation*. Springer, 1997.

[72] Ken Schwaber. *Agile project management with Scrum*. Microsoft press, 2004.

[73] Ken Schwaber and Jeff Sutherland. The scrum guide. *Scrum Alliance*, 21, 2011.

[74] Scrum.org. https://www.scrum.org/resources/what-is-scrum, November 2017.

[75] Andy Singleton. https://andysingleton.com/agile-2014-appearance-for-maxos-the-new-continuous-agile-3f4b3f564801, October 2017.

[76] Andy Singleton. http://www.continuousagile.com/unblock/ea_matrix.html, October 2017.

[77] Jennifer Stapleton. *DSDM, dynamic systems development method: the method in practice*. Cambridge University Press, 1997.

[78] Jennifer Stapleton. Dsdm: Dynamic systems development method. In *Technology of Object-Oriented Languages and Systems, 1999. Proceedings of*, pages 406–406. IEEE, 1999.

[79] Jennifer Stapleton. *DSDM: Business focused development*. Pearson Education, 2003.

[80] Jeff Sutherland. Agile can scale: Inventing and reinventing scrum in five companies. *Cutter IT journal*, 14(12):5–11, 2001.

[81] CA Technologies. https://www.ca.com/us/company/newsroom/press-releases/2015/ca-technologies-most-businesses-will-be-software-driven-in-36-months.html, November 2017.

[82] K Thompsom. Recipes for agile governance in the enterprise, 2013.

[83] 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.

[84] Satisha Venkataramaiah. https://confengine.com/agile-india-2016/proposal/1677/lean-enterprise-agile-framework-a-systems-thinking-approach-to-scale-deliverables, October 2017.

[85] VersionOne. 11th annual state of agile survey. Technical report, VersionOne Inc., 2017.

[86] Jan vom Brocke, Alexander Simons, Bjoern Niehaves, Kai Riemer, Ralf Plattfaut, and Anne Cleven. Reconstructing the giant: On the importance of rigour in documenting the literature search process. In *ECIS*, volume 9, pages 2206–2217. ECIS 2009 Proceedings. 161, 2009.

[87] Don Wells. http://www.extremeprogramming.org, November 2017.

[88] ZilicusPM. http://blog.zilicus.com/wordpress/kanban-board-visualize-project-work-workflow/, November 2017.