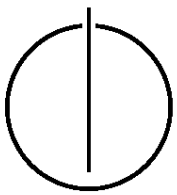
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Identifying recurring Challenges and Best Practices of Agile Coaches and Scrum Masters and Documenting them as a part of a Large-Scale Agile Development Pattern Language

Nina-Mareike Harders



FAKULTÄT FÜR INFORMATIK

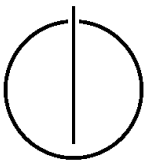
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Identifying recurring Challenges and Best Practices of Agile Coaches and Scrum Masters and Documenting them as a part of a Large-Scale Agile Development Pattern Language

Identifizierung wiederkehrender Herausforderungen und bewährter Praktiken von agilen Trainern und Scrum Mastern und deren Dokumentation als Teil einer Mustersprache für Large-Scale Agile Development

Author:	Nina-Mareike Harders
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Ömer Uludağ, M. Sc.
Date:	May 15, 2019



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

Munich, May 27, 2019 Nina-Mareike Harders

Abstract

With the success of agile methods for small teams, large IT organizations and entire enterprises become increasingly interested in applying these methods at scale. While small teams already face a variety of challenges coming with agile methods, there are even more impediments to overcome in large-scale agile methods. These challenges include among others coordinating geographically distributed teams, or managing interdependencies across multiple teams working on the same product. While research on challenges in large-scale agile development is available, publications on best practices to encounter them is scarce. One way to document such best practices in a structured manner is to use pattern languages. Pattern languages are collections of patterns that focus on the same domain, while each individual pattern describes a practice-proven solution to a recurring problem within a given context. Even though there are pattern languages focusing on agile development, as well as distributed project management, none solely concentrates on large-scale agile development, raising the demand for a large-scale agile development pattern language. Most pattern languages do not provide any structural filter, leaving the user with a set of patterns, without knowing how to search for a suitable one. To solve this problem, this thesis proposes a pattern language, which divides patterns into five categories: coordination patterns, methodology patterns, viewpoint patterns, anti-patterns, and principles. Besides, a set of stakeholder and concerns are defined to provide a filter mechanism to the user. This structure allows the user to first filter for his specific role and problems and find suitable patterns accordingly. This thesis uses a mixed-methods approach, consisting of a Design Science Approach and Pattern-Based Research. By conducting fourteen interviews with industry experts we evaluated an initial proposal of the pattern language. In a second step, patterns and recurring concerns for the stakeholders Agile Coach and Scrum Master have been identified by conducting thirteen semi-structured interviews. Overall, 76 pattern candidates have been identified, which resulted by the application of the rule of three in two coordination patterns, four methodology patterns, two viewpoint patterns, five principles, and two anti-patterns. This work is part of a major research project aiming at establishing a comprehensive pattern catalog for large-scale agile development.

Contents

List of Figures

List of Tables

List of Abbreviations

ART	Agile Release Train
C	Concern
CO-Pattern	Coordination Pattern
DV	Developer
EA	Enterprise Architect
EAM	Enterprise Architecture Management
EuroPLoP	European Conference on Pattern Languages of Programs
ID	Identifier
ISO	International Standardization Organization
IT	Information Technology
KPI	Key Performance Indicator
LeSS	Large-Scale Scrum
LSAD	Large-Scale Agile Development
LSADPL	Large-Scale Agile Development Pattern Language
M-Pattern	Methodology Pattern
No.	Number
OKR	Objectives and Key Results
PA	Platform Architect
PBRD	Pattern-Based Research Design
PLoP	Pattern Languages of Programs

PM	Project Manager
PO	Product Owner
PRINCE2	Projects in Controlled Environments 2
R&D	Research & Development
RA	Requirements Area
RAGE	Recipes for Agile Governance in the Enterprise
RQ	Research Question
SA	Solution Architect
SAFe®	Scaled Agile Framework
UML	Unified Modeling Language
UX	User Experience
V-Pattern	Viewpoint Pattern
WIP	Work in Progress
XP	Extreme Programming

Outline of the Thesis

CHAPTER 1: INTRODUCTION

This chapter introduces the topic and motivation for this thesis. It presents a short overview about the research approach and general outline.

CHAPTER 2: FOUNDATIONS

All relevant terms and concepts for this work are explained in this chapter, such as agile development, large-scale agile development, patterns, pattern languages, pattern catalog, as well as the responsibilities and challenges of Scrum Masters and Agile Coaches.

CHAPTER 3: RELATED WORK

The first part of this chapter describes literature related to large-scale agile development, while the second part examines different organizational and agile pattern languages.

CHAPTER 4: EVOLVING THE LARGE-SCALE AGILE DEVELOPMENT PATTERN LANGUAGE

Divided into two development iterations, this chapter explains the development of a large-scale agile development pattern language based on the evaluation with fourteen industry partner. The results of the evaluation will be presented in detail.

CHAPTER 5: IDENTIFICATION OF RECURRING CONCERNS AND BEST PRACTICES

This chapter first explains the methodology of how the recurring challenges and patterns have been identified. Afterward, all concerns are described in detail and five exemplary patterns are presented.

CHAPTER 6: DISCUSSION

Key findings of this work are discussed on limitations are pointed out.

CHAPTER 7: CONCLUSION

This concluding chapter summarizes the findings and proposes approaches to future work.

1. Introduction

1.1. Motivation

Not only do agile methods promise higher customer satisfaction by higher quality and better products, but also enable a better and fast reaction to changing requirements [28]. Instead of focusing on comprehensive planning and following a predefined process such as the Waterfall Model proposes, agile methods like Scrum concentrate only on a short time horizon. Active customer involvement and frequent feedback loops are the core of all agile frameworks [27]. However, agile practices are only suitable for small, co-located, self-organizing teams with at most 50 people, working on non-critical projects [85]. Still, enterprises and large IT organizations are motivated by the success of agile methodologies and scale agility according to their needs [27]. This results in multiple agile teams working on the same product or even distributed agile teams [66]. Benefits for adopting agile include among others the ability to manage changing priorities, increasing the productivity of teams, higher software quality as well as project cost and risk reduction [80]. However, large-scale agile development brings not also benefits, but also a large amount of challenges [26, 78]. Those challenges involve problematic, non-agile mindsets, resistance to change, missing support by management, and missing skills [26, 78, 80]. Even though there is much research on challenges in large-scale agile development, the literature on practices to tackle these challenges is scarce. Dikert et al. [26] identified success factors to be considered such as piloting, the provision of training and concentration on agile values. However, concrete practices that can be adopted in practice to encounter a certain challenge are missing. Dikert et al. [26] also stress the demand for extensive research on scaling practices adopted in practice to counter the identified challenges. One way to document such problem-solution pairs are patterns, which are proven solutions to recurring problems in a given context [18]. Moreover, a collection of patterns encountering problems from the same domain is called a pattern language. Each pattern consists of a problem section, where the problem is elaborated, and a solution-section, containing a detailed description of the solution and steps to adapt the solution in practice. Even though several pattern languages are focusing on agile development [8, 20, 31, 60] and even distributed project management [71, 79], a pattern language purely focusing on large-scale agile development does not exist. Compared to small agile teams, large-scale agile organizations include a significant number of additional stakeholders to be considered, e.g., Program Manager, Portfolio Manager, or Agile Coaches [78]. Therefore, dividing the pattern language into patterns for different stakeholders increases the usability of the language. [78] also serves as the primary source for challenges for this research. 22 out of 79 overall iden-

tified challenges in large-scale agile development refer to cultural, mindset-related, and communication-related problems. Most of these challenges refer to the stakeholders *Agile Coach* and *Scrum Master*. Therefore, this work focuses on identifying patterns for these two stakeholders.

1.2. Research Objective

This thesis aims at introducing a pattern language for large-scale agile development and identify first patterns for Agile Coaches and Scrum Masters. Therefore, this work defines the following three research questions.

> **Research Question 1:** What are recurring challenges of Agile Coaches and Scrum Masters in large-scale agile development?

A mentioned before, there are publications listing a large number of challenges for all kinds of stakeholders [26, 78]. However, a validation of practical relevance is missing. Therefore, the purpose of Research Question (RQ) 1 is to examine, which challenges of Agile Coaches and Scrum Masters identified in literature occur in practice. As it is the nature of patterns to solve frequent problems, this question further evaluates, which challenges recur through several organizations. The goal is to develop a list of recurring concerns in large-scale agile development for Agile Coaches and Scrum Masters.

> **Research Question 2:** What are best practices for addressing recurring challenges of Agile Coaches and Scrum Master in large-scale agile development?

Based on the identified concerns in RQ1, best practices for addressing those are the artifact of interest for RQ2. Best practices can be either meetings, methods, visualizations, or principles used by the Agile Coach or Scrum Master in order to address the concern. To answer this question, a list of practices is developed and examined for recurrence. If a practice is observed at least three times, it will be included as a pattern in RQ3. Otherwise, it remains a pattern candidate.

> **Research Question 3:** How can these best practices be documented as part of a large-scale agile development pattern language?

In a first step, an initial version of the large-scale agile development pattern language is developed based on the findings by [78] and the structure of the Enterprise Architecture Management Pattern Language [49]. Afterward, the pattern language is evaluated by industry experts and based on the feedback a final version is developed. Finally, the results from RQ1 and RQ2 are documented in the form of patterns for the Large-Scale Agile Development Pattern Language (LSADPL).

1.3. Research Approach

This thesis uses a mixed-methods approach [11], consisting of the design science research methodology [41, 64] and the pattern-based research design [13]. The goal of the design science approach is to create a new artifact that on the one side is *relevant* to the environment in which it is used. On the other side, the artifact must be created on a sound knowledge base to increase the validity of the artifact. This is referred to as *rigor*. Moreover, the artifact must be evaluated and refined using scientific methods. The approach is illustrated in Figure 1.1. The artifact developed in this work is the large-scale agile development pattern language and is assessed by fourteen structured interviews with industry experts, who are all experienced in large-scale agile development projects. This evaluation allows incorporating a great number of business needs, consequently increasing the relevance for the industry. Additionally, a literature review on patterns, pattern languages, and large-scale agile development ensures the theoretical foundation of the pattern language.

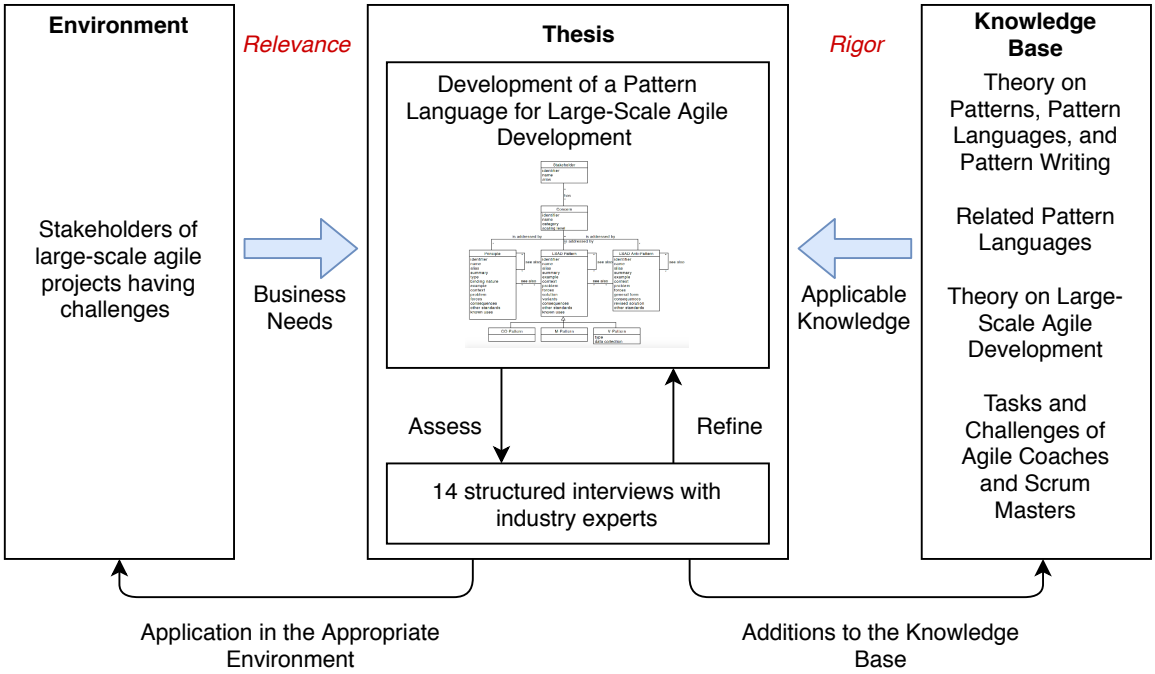


Figure 1.1.: Scientific Approach for designing the Large-Scale Agile Development Pattern Language based on [41]

The second scientific approach has been developed by Buckl et al. [13] and is a refinement of the design science approach [41], focusing on patterns as the primary artifact. Figure 1.2 illustrates the process of the pattern-based research methodology and shows,

which parts of it are incorporated for this thesis. Pattern-Based Research Design (PBRD) takes both, literature and observations in practice to develop a set of pattern candidates. This work uses the rule of three [18] to make a pattern candidate an actual pattern. A best practice requires a minimum of three observations in different organization to be considered a pattern. The same rule is applied for examining the recurrence of challenges. 13 semi-structured interviews with Agile Coaches and Scrum Masters serve as the primary data source for the patterns, in which both, recurring concerns and best practices have been inspected. PBRD also suggests applying the patterns in practice to observe deviations between documented patterns and their adoption in practice. These insights can be further used to refine existing patterns. Due to the limited time frame, the evaluation of the identified patterns is out of scope for this work.

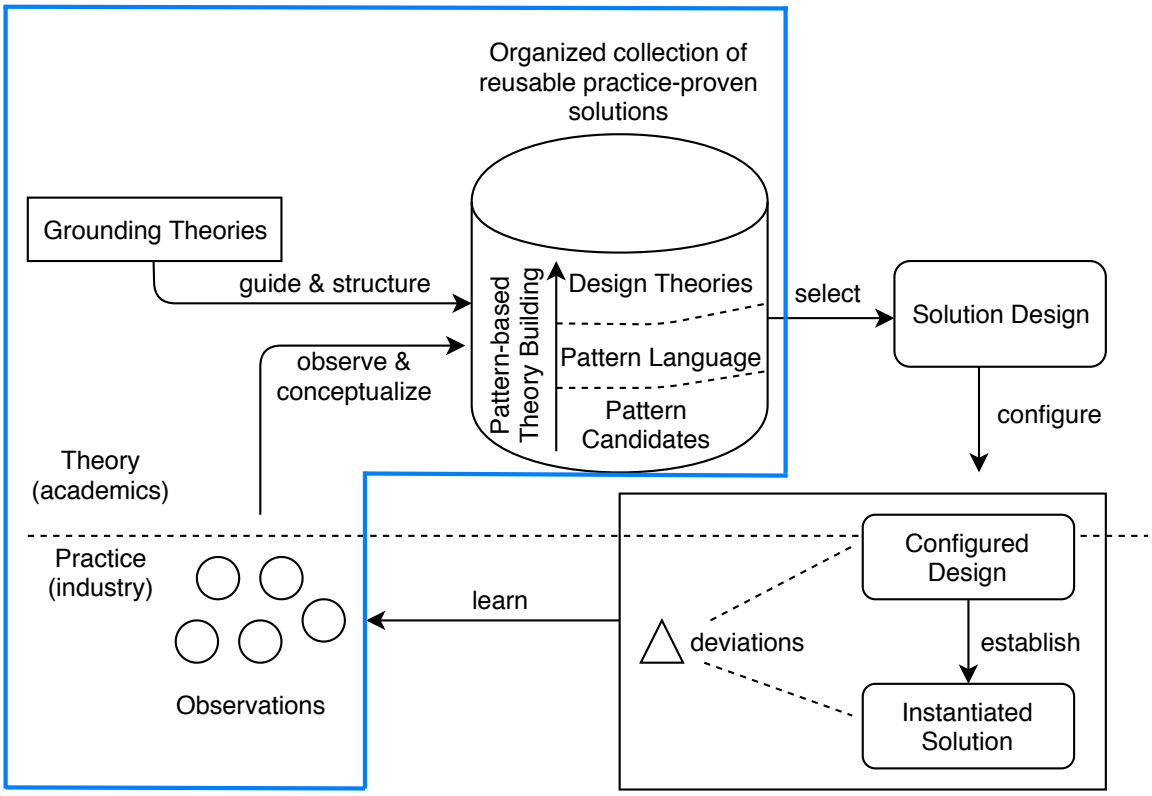


Figure 1.2.: Pattern-based Research Design based on [13]

The thesis is structured as follows. Section 2 introduces essential terms and concepts used throughout this work such as agile and large-scale agile development, responsibilities and challenges of Scrum Master and Agile Coaches, and the concept of patterns. Section 3 summarizes to most related work regarding challenges in large-scale agile devel-

opment and illustrates related pattern languages. Based on the previous findings, Section 4 describes the entire development process of a pattern language for large-scale agile development, including an evaluation of the pattern language. After that, the first part of Section 5 lists all challenges and concerns identified in practice and analyses them for recurrence. The second part describes all identified practices and describes five exemplary patterns in detail. Key findings and limitations are elaborated in Section 6. Finally, Section 7 summarizes the results and provides an outlook for future research.

2. Foundations

This chapter will explain all relevant terms and concepts used in this thesis. It will first define the concepts and principles of classic agile development and further describe the major framework, Scrum, for this domain. In contrast, the definition, concept, and challenges of Large-Scale Agile Development will be explained afterward. The chapter will describe two popular frameworks for scaled agile: Large-Scale Scrum (LeSS) and SAFe. Besides, two major stakeholders of agile teams will be explained as they build the scope of this thesis. Finally, the terms of pattern, pattern language and pattern catalog will be defined individually. A section about good pattern writing practice will conclude this chapter.

2.1. Agile Development

2.1.1. Definition

In contrast to the traditional software development models, like the waterfall model, which heavily focus on processes and methods [10], the concept of 'Agile' consists of a set of principles and a set of values. Moreover, agile practices are defined steps to be taken, based on these principles and values [61, 68]. The goal of agile development is to deliver software as fast as possible and provide the highest possible value to the customer during a short development cycle [57]. In the traditional models, all requirements are set, and decisions are made during the design stage, which would not allow for any change at the later point in the project without major impact. This lead to many big projects' failure [10]. Agile practices try to solve this issue by being more resistant to change. The development is incremental, which means only features with a high priority for the customer are developed within the next development cycle [57]. Because there is no fixed schedule of when to implement which functionality, changing requirements and circumstances do not have a large impact on agile projects in comparison to traditional project management approaches. The theoretical foundation for agile development has been documented in the 'Agile Manifesto'. It has been published 2001 by seventeen authors [6] and states twelve agile principles and four agile values. The four values are:

1. **Individuals and interactions over processes and tools**: good processes and tools can not generate value to the customer if the team members are not motivated and have the right mindset [58]

2. **Working software over comprehensive documentation**: the key point here is a 'fit-for-purpose' documentation [58] as well as delivering value to the customer
3. **Customer collaboration over contract negotiation**: the relationship between customer and supplier is, and both stakeholders have to understand that a good product can only be produced by good and continuous collaboration. It is more important to communicate than to have tight and long contracts [58]
4. **Responding to change over following a plan**: while there is a plan, it has to be able to adapt to change and consequently needs to be very flexible [58]

While processes, tools, documentation, contract negotiation and following a plan are indeed important artifacts and required in any project, agile methods rely more on individuals, working software, customer collaboration and responding to change.
The other part of the Agile Manifesto defines twelve agile principles which focus on the following key aspects [6]:

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

These principles concentrate on customer-value-driven development and individuals' mindsets. Typically, agile teams consist of four to ten co-located people [57]. As mentioned before, the agile principles and values form the basis for all agile practices and agile methods [68]. People can apply agile methods, but without an agile mindset, the outcome will not be as expected [57].

2.1.2. Agile and Lean Development

Lean is a term often mentioned in the same breath with agile development. It origins in the car manufacturing domain and was introduced by Toyota in 'The Toyota Way' [19]. A synonym for lean is 'minimal' [19] as the focus is to reduce waste as much as possible and establish a culture of continuous improvement. Similar to the concept of agility, lean also defines principles, but no clear practices [65]. Lean principles include the elimination of waste, fast delivery, decisions should be made as late as possible, team empowering and seeing the big picture [65]. Five major lean concepts complement these principles [81]:

1. **Value** is defined by the customer.

2. A **value stream** is defined for a process and describes how much value a process step adds or removes.

3. The production process **flows** continuously.

4. Manufacturing is realized by a **Pull** principle, which means production is only triggered if a customer orders a product.

5. Everyone should aim for **perfection** by reducing waste and maximizing value.

These concepts are very similar to agile development as the ultimate goal is also to provide value to the customer while continuously improving the development process and team communication [57]. However, [19] identified several aspects in which lean and agile differ. While agile relies on a Do-Inspect-Adapt principle, lean bases on an Inspect-Plan-Do approach. Moreover, lean focuses heavily on processes and teams as the lowest possible unit, whereas agile concentrates on people, seeing individuals and interactions as the smallest units. This is also reflected in the first value of the agile manifesto [6].
Wang et al. [81] surveyed how agile and lean approaches are combined in software development. They found six types of combination: non-purposeful combination (1), using lean approaches for communicating with other business units but using agile within the team (2), lean approach for adopting agile approaches (3), using lean to improve agile (4), using lean to transform agile (5) and finally, using agile and lean in parallel (6). The most frequently used combination is the fourth type: using lean to improve agile [81].

2.1.3. Scrum

An example of an agile software development framework is Scrum, which is one of the most popular agile frameworks currently in use [10]. The official Scrum Guide by Ken Schwaber and Jeff Sutherland describes Scrum as 'a framework within which people can address adaptive problems, while productively and creatively delivering products of the highest possible value' [70], while Wirdemann & Mainusch define Scrum as 'a framework for managing complex projects' [86]. As projects face unexpected change, the project needs to be structured in a way that it is adaptable to change [86]. Scrum is a framework, meaning that it defines the outline of the project management, but does not define clear methods to be used during meetings or the development phase [86]. Consequently, it defines a set of roles, events, and artifacts, that must be part of every project that uses Scrum.
The main intention of Scrum is not to plan the product in detail in the beginning, but to only define the basic functionality. This functionality is then incrementally developed within many Scrum iterations [35]. The ideal Scrum team is small and co-located, which means they are not distributed across different sides. The principle of self-organization is one of the critical foundations of Scrum, as the team can fully decide on which work packages and tasks to work on, without being instructed by a manager [35].
Table 2.1 shows the essential roles, meetings, and artifacts defined in the official Scrum Guide [70]. There can be more roles (e.g., the customer, or the end user [34]) and more specific events (see the 'Estimation Meeting' explained in [35]). In the following, this chapter will only characterize the central roles, events, and artifacts provided by the official Scrum Guide.

Roles	**Events**	**Artifacts**
Scrum Master	Sprint	Product Backlog
Product Owner (PO)	Sprint Planning	Sprint Backlog
Development Team	Daily Scrum	Increment
	Sprint Review	
	Sprint Retrospective	

Table 2.1.: Scrum Roles, Events and Artifacts

Scrum Roles. There are three roles in every Scrum team: the Scrum Master, the Product Owner, and the Development Team [70]. The Product Owner maintains the Product Backlog and is responsible for prioritizing and clarifying backlog items. Further, he is accountable for all financial matters of the project [34, 70]. On the other side, the Scrum Master facilitates the Scrum process, consequently assuring that Scrum practices and principles are correctly applied. Whenever problems arise, the Scrum Master is responsible for fixing them [34]. Finally, the Development Team is in charge of creating a deliverable, the so-called *Increment* [70].
Scrum Events. The five main events in Scrum are represented in Figure 2.1. In Scrum, a development iteration is called in *Sprint*, which is typically one to four weeks long [86].

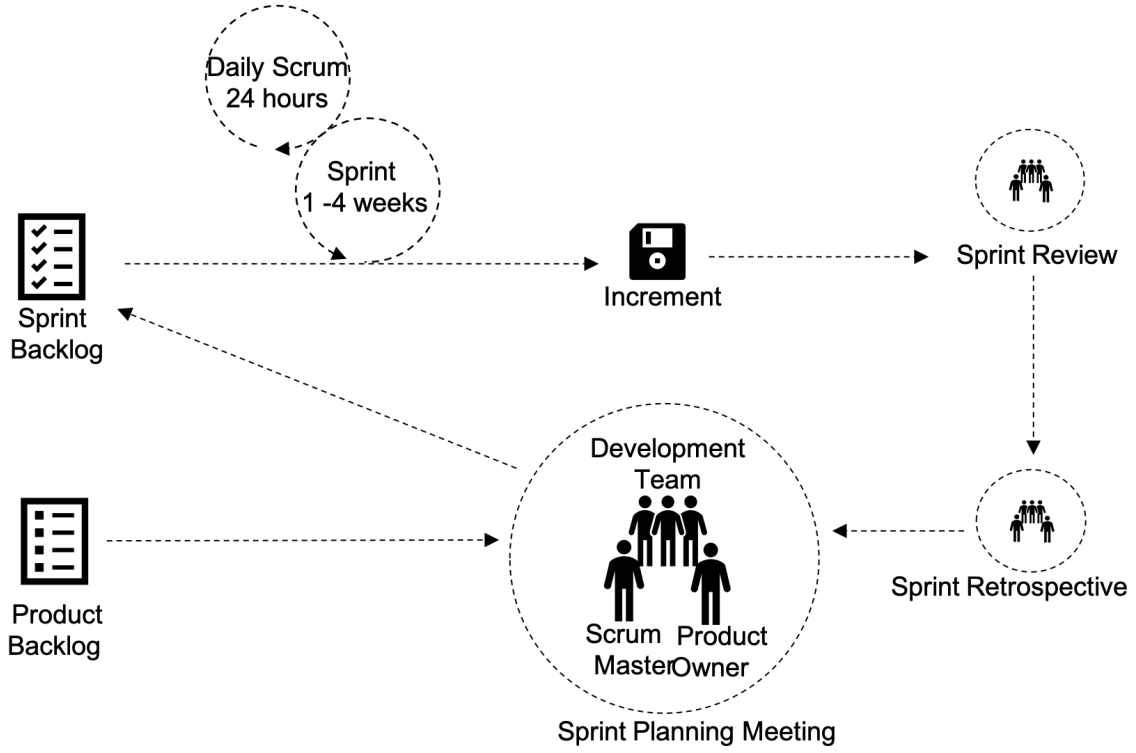


Figure 2.1.: The overall Scrum Framework based on [86]

Before a Sprint, the Scrum Team meets for a *Sprint Planning Meeting*, where the Sprint Backlog is defined based on the Product Backlog. During a Sprint, the team meets every morning and performs the *Daily Scrum*, a short informal meeting, where every team member explains what he has achieved since the last *Daily Scrum*, what he will do until the next Daily and if he has any impediments that require action [70]. At the end of each sprint, first a Sprint review is carried out, where the *Increment* is evaluated and the Product Backlog is adapted. While the *Sprint Review* focuses on the Increment, the *Sprint Retrospective's* focal point is the relationship between the members of a Scrum Team and how to improve the teamwork [34].

Scrum Artifacts. The framework concludes with three types of backlogs that have been mentioned several times before. First, the product backlog is an ordered list of features, requirements, enhancements, changes or functions that are required for the product. The Product Backlog evolves over the project duration as it is constantly updated by the Product Owner whenever changes occur [70]. In contrast to the Product Backlog, which lasts the whole project, a Sprint Backlog is only defined for one unique Sprint. It contains all Product Backlog items that are planned to be implemented during the Sprint. The Backlog items are part of the resulting Increment [34], which is a usable version of the product [70].

It does not need to be released after the Sprint, but it is a step closer to the final product [70].

2.2. Large-Scale Agile Development

Hereafter, a definition of the phrase *Large-Scale Agile Development* will be given, and two essential scaling-agile frameworks will be presented: Large Scale Scrum (LeSS) and the Scaled Agile Framework (SAFe).

2.2.1. Definition

While the original agile methods are best suited for small, co-located teams, their success has motivated larger companies to adopt them for the use on a scaled level [27] for more complex projects [57]. There is no clear, generally-agreed definition on the term *large-scale*, however Dingsøyr et al. [29] define a taxonomy of scale stating *large-scale* requires 2 - 9 agile teams working together [29]. According to the taxonomy, more than ten agile teams already apply to the term *very large-scale*. In contrast to this definition, the one by Dikert et al. [26] requires more than 50 people or at least six teams working together to be considered *large-scale*. Larman and Vodde [53] describe their experience with *large* agile projects. They had around 800 people involved, distributed onto five sites and about 15 million lines of code. All these definitions show that scale is mostly driven by team size and the number of people involved. This is also proven by Dingsøyr and Moe [27] in a workshop asking people for their definition of the term *large* in the context of agile development. Power [66] brings up another important point and describes three types of levels in a scaled agile organization: first, agile approaches used in a large organization as a team, meaning independent teams using agile methods, which is not the scope of *scaled agile*. Second, agile approaches used in a large development effort in a large company and third, organizational agility. This thesis focuses on the second point, where a large number of people have to work together in an agile manner. This requires the scaling of traditional agile methods. Organizational agility describes the mixture of informal networks in a formal organizational structure, resulting in more trust and faster reaction to change [52]. This aspect is indeed important for an organization using scaled-agile approaches; however, it is not the main point of *large-scale agile development*.
Summarized, this thesis uses the following definition of large-scale agile development:

> Large-Scale Agile Development uses agile methods, principles and values applied to at least two distinct agile teams, working together on the same project or product.

We do not make assumptions on the location of the teams, as they can be co-located or distributed and still require scaled agile approaches. Even though all teams work on their tasks on their own, there is still a need for communication and coordination across all

teams, for example, the distribution of User Stories, dependencies across functionality, and many more.

2.2.2. LeSS - Large-Scale Scrum

LeSS is one of the best-known frameworks for agile methods at scale. It is 'Scrum applied to many teams working together on one product' [54]. The founders of the LeSS, Craig Larman and Bas Vodde, define two frameworks, namely LeSS and LeSS Huge. The first fits two to eight individual teams, while the later works well with more than eight teams and thousands of people working together on the same product.

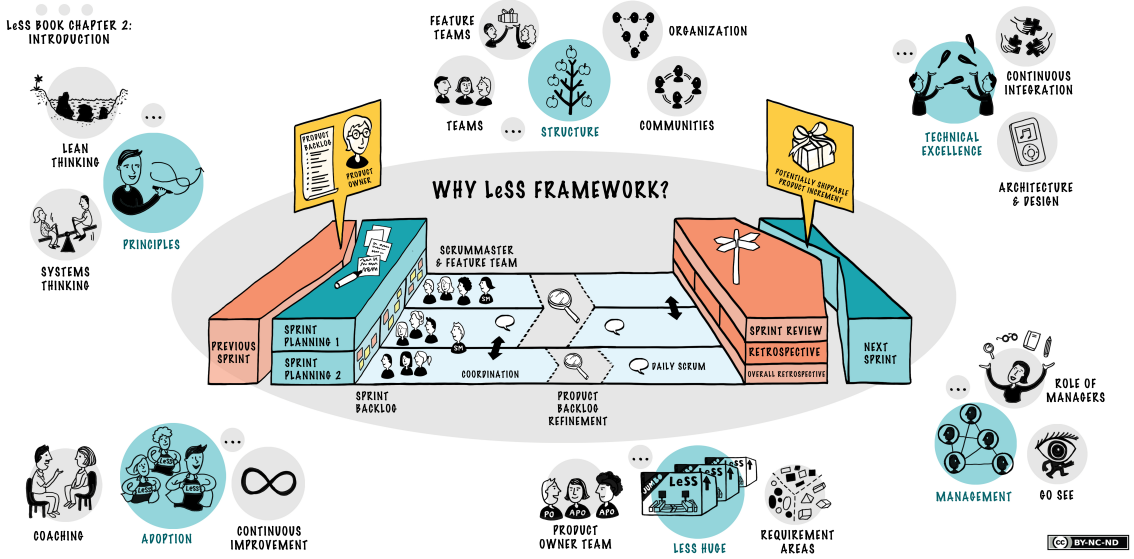


Figure 2.2.: Large-Scale Scrum by [16]

LeSS. Similar to Scrum, there is exactly one Product Owner, one Product Backlog and one Increment. However, several teams are having each a Scrum Master and a Development Team. All teams work within the same sprint, meaning they start and end their sprint at the same time, have a common Sprint Planning, Review and Retrospective [54]. The major difference between traditional Scrum and LeSS lies in Sprint Planning, which is split into two phases: Sprint Planning One and Sprint Planning Two. During the first Sprint Planning, the Product Owner and either whole teams or (rotating) representatives of teams meet and allocate items from the Product Backlog to teams. In Sprint Planning Two, either each team has its own Sprint Planning based on the allocated backlog items, or several teams perform their Sprint Planning together because their items are closely related. Larman and Vodde [54] emphasize that continuous integration is essential for all teams during a Sprint 'as a way to inform and support their communication'. Just like

in traditional Scrum, a Sprint is closed with a Retrospective. In LeSS, the Product Manager, all Scrum Master, Managers (if any) and team representatives attend a Retrospective, discussing cross-team and system-wide impediments and issues. Retrospectives on team-level are also mandatory in the LeSS framework. Additionally, an Overall Sprint Review closes the Sprint [54]. In LeSS it is substantial that the development team is a *Feature Team*, meaning they are 'true cross-functional, cross-component full stack teams' [54]. All meetings, roles, and artifacts are shown in Figure 2.2.

LeSS Huge. LeSS Huge is sometimes referred to as a 'stack of LeSS' [54]. It is very similar to the 'small' LeSS, as it also includes one common Product Backlog and Sprint synchronization across all teams. However, due to a large number of teams and people, it is not possible to have a two-tier hierarchy, like in LeSS. Therefore, teams need to be divided into a larger hierarchy across different *Requirements Area (RA)* [54]. Each requirement area is managed by an individual *Area Product Owner*, who focuses on the Area Product Backlog. Four to eight *Area Feature Teams* are assigned to one Area Product Owner, resulting in a LeSS-like work environment. Eventually, there is one overall Product Owner, who focuses on complete product development and optimization. Additionally, he is responsible for assigning Area Product Owners and slicing the product into requirement areas [54].

LeSS also defines several principles, which people working in LeSS should adhere to, in order to achieve the best results. The LeSS principles are as follows [54]:

- **Large-Scale Scrum is Scrum**: the elements of Scrum should only be applied to a larger context
- **Transparency**: includes a definition of 'done', collaboration and communication
- **More with less**: LeSS does not introduce more roles and processes, instead reallocates responsibilities
- **Whole-product focus**: achieved by only maintaining one Product Backlog by one single Product Owner
- **Customer-centric**: all team members need to focus on creating value for the paying customer
- **Continuous improvement towards perfection**: everyone needs to try to improve the product constantly
- **Lean thinking**: managers should not delegate work, but rather act as teachers, who facilitate a lean mindset within the organization
- **Systems thinking**: everyone has to think of the system as a whole, but in parts or subsystems, as this is what the customer does
- **Empirical process control**: rather than just applying best practices, everyone should constantly review their ways of working and try to improve according to the context

- **Queuing theory**: figure out how 'systems with queues behave in the Research & Development (R&D) domain' [54]

2.2.3. SAFe® - Scaled Agile Framework

The Scaled Agile Framework (SAFe®) the most popular scaling agile framework [80], promising not only happier and more motivated people and up to 50% increased productivity, but also up to 75% faster time to market and more than 50% defect reduction [50]. SAFe® combines three approaches: agile development, systems thinking, and lean product development. There are three configurations, which differ in the scaling factor. Essential SAFe® serves as a starting point for the implementation of large-scale agile practices. Three-level SAFe® suits best for scaling agile to up to five to ten individual teams, as well as the development of different solutions [50]. Finally, four-level SAFe®, presented in Figure 2.3, is designed for the integration of an entire organization, hence providing the greatest scaling factor. The bottom layer handles individual agile teams, consisting of Scrum Master, Product Owner, Development Team, and other required enterprise-specific roles. The practices on this layer can vary between all kinds of small-scale agile frameworks, like Scrum, Extreme Programming, or Kanban. Instead of calling an iteration a 'Sprint', SAFe® calls it a *Cadence*. Next, the program layer contains the *Agile Release Train (ART)*, which is a 'self-managing and organizing team-of-Agile-teams that plans, commits, and executes together' [50], typically five to twelve teams large [50]. An ART describes a solution that continuously requires development effort and achieves high efficiency by the implementation of lean-agile principles such as continuous integration. The cadences of all teams within an ART are synchronized, i.e., have the same start and end date. The third level is called the Value Stream Level and consists of one *Value Stream*, which either provides value to the customer or other internal processes [43]. A value stream consists of several ARTs, which are also synchronized [50]. Further, *epics* are initiatives on the value stream level, which take several Program Increments to develop. Finally, the portfolio level aggregates several value streams and is the top level of SAFe®. Budgeting of value streams, as well as business objectives, are managed on this level [50].
SAFe® includes not only artifacts and practices but also a set of values and principles. It defines four values: alignment, built-in quality, transparency, and program execution. *Alignment* targets the people's mindset, wanting them to think of the organization as a whole and not in different teams. This is reached by synchronization of cadences and joint meetings, like the Program Increment Meeting. *Built-in quality* refers to a minimum level of quality within the solutions to prevent later programs. *Transparency* is achieved by 'sharing progress and facts openly across all levels' [50], which is done by the 'Inspect & Adapt' event as well as measurements [50]. Finally, *Program execution* focuses on teams being able to work in order to produce outcomes and value.
In addition to the four values, SAFe® describes nine principles that should always be applied: (1) take an economic view, (2) apply systems thinking, (3) assume variability; preserve options (4) build incrementally with fast, integrated learning cycles, (5) base mile-

stones on objective evaluation of working systems, (6) visualize and limit WIP, reduce batch sizes, and manage queue lengths, (7) apply cadence, synchronize with cross-domain planning, (8) unlock the intrinsic motivation of knowledge workers, and (9) decentralize decision-making. An organization should stick to all of these principles, independent of the degree of adoption of SAFe® [50].

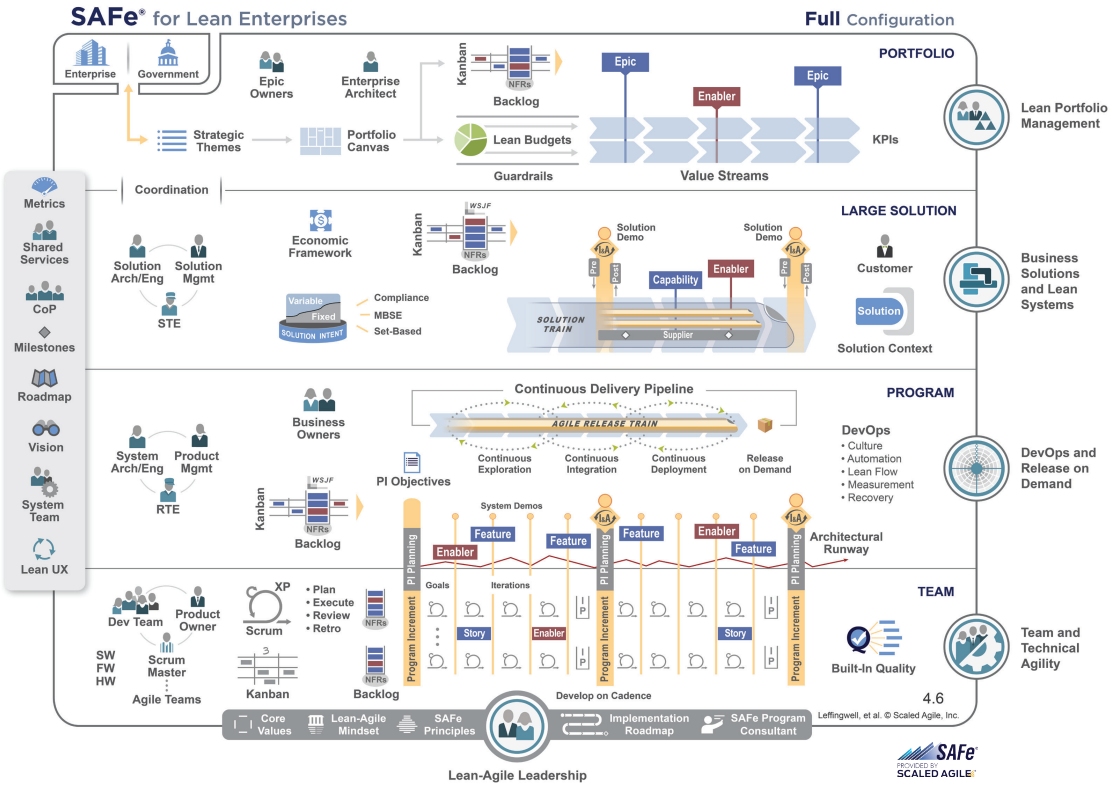


Figure 2.3.: Big Picture of Four-Level SAFe® by [42]

2.2.4. Summary and other Frameworks

The previous sections provided a short overview of the differences between small-scale and large-scale agile development and summarized three major agile frameworks: Scrum, Large-Scale Scrum, and the Scaled-Agile Framework. However, there are more scaling-agile frameworks, such as the Spotify Model [51], where the organization is divided into chapters, guilds, and tribes, or the Disciplined Agile Delivery [1], which extends agile practices such as Scrum, XP, Kanban, or Test-Driven Development. Similar, Nexus [69]

builds upon Scrum and focuses on solving challenges with inter-dependencies between multiple Scrum teams [4]. Other frameworks include RAGE [76], Scrum of Scrums [44], and Enterprise Scrum [36]. A comprehensive comparison of the frameworks can be found in [4].

2.3. Stakeholders

This thesis focuses on the recurring challenges of two stakeholders: Scrum Master and Agile Coaches. The following sections concentrate on introducing both stakeholders' responsibilities and tasks within an agile setting.

2.3.1. Scrum Master

Responsibilities

The Scrum Master has been introduced as a Scrum role in chapter 2.1.3. He is responsible for the success and productively of a Scrum team [86] as well as the correct adoption of all Scrum principles, values, and practices [70]. The Scrum Master acts as a coach for the development team and Product Owner, teaching them Scrum and showing them methods to become more efficient [70]. Moreover, he removes impediments [62, 70, 86], prevents the team from any harmful external influence [86] and facilitates the Scrum Events. Because Scrum Teams are self-organized, the Scrum Master is neither a line manager [5] nor a project manager with the power to assign tasks to team members and to allocate and direct work [86]. Instead, he operates as a *Servant Leader*, who does everything for the team [70]. In close cooperation with other Scrum Masters, he shapes the agility of an organization to increase the efficiency of the overall organization [70]. Diebold et al. [25] examined the adoption of Scrum roles in practices and found that being a Scrum Master is a full-time job at the beginning of a project, transforming into a half-time job once the project stabilized. Most Scrum Masters partly act as another role, e.g., as Developer [25, 62]. Additionally, a Scrum Master who is also a developer, gains more respect by the development team, because he can understand their impediments and tasks well [25, 86]. In a scaled agile context, the tasks remain the same but are extended by the cooperation with other Scrum teams within a program and the integration of each Scrum team's code [5].

Challenges

A Scrum Master faces several challenges, which have been identified by Uludağ et al. [78], who conducted a structured literature review to identify recurring challenges in a large-scale agile setting. They identified ten challenges, mostly dealing with the geographical distribution of several agile teams or even single team members, or dealing with communication issues. The ten challenges are:

- Coordination geographically distributed agile teams

- Facilitating agile teams to participate at cross-shore meetings
- Synchronizing working hours of cross-shore agile teams
- Dealing with lacking team cohesion at different locations
- Building trust of stakeholders in agile practices
- Establish a culture of continuous improvement
- Rearranging physical spaces
- Dealing with higher-level management interference
- Dealing with cultural differences between cross-shore agile teams
- Synchronizing sprints in the large-scale agile development

However, some of these challenges also apply to the traditional, small-scale agile approach.

2.3.2. Agile Coach

Responsibilities

In contrast to the Scrum Master, who is an explicitly defined role within the Scrum framework, there is no generally agreed definition for the role of the Agile Coach. Fraser et al. [33] explain, he is 'often an itinerant who may observe, mentor, negotiate, influence, lead, and/or architect everything from team organization to system architecture' [33]. It is an Agile Coach's primary task to help the team reflect their work and provide guidance [55], rather than manage a team and remove impediments [33]. On the one hand, an Agile Coach needs technical skills to understand what the team is doing [67], and on the other hand, he needs strong social skills in order to understand the social relationships within the team [33, 67]. Additionally, his social skills enable him to understand the psychology of individuals and how this influences the team as a whole [33]. Among training a team or an organization in agile practices, showing new possibilities and providing feedback, the most crucial task of an Agile Coach is to establish a culture of learning [33]. Davies and Sedley [23] summarize the tasks as follows: observe the team, provide feedback, train and educate, moderate and support. Kelly [47] describes two approaches to coaching. In the *directive* approach, the coach has extensive knowledge about the domain and mostly teaches a team in applying practices. In contrast, the *non-directive* approach does not require knowledge about agile practices. The focus lies on the coaching itself: the coach helps the team to grow on their own and increase their performance. The latter approach is more suitable to teams that already are familiar with agile practices, while the first approach fits new teams [47]. Agile Coaches can either be internal employees or external

consultants [67]. Both options have their benefits: while an internal coach knows the environments and circumstances, an external consultant can provide a fresh view and generate approaches very different from the current ones [33]. A combination of internal and external coaches working together as a team works best [33, 55].

Challenges

Uludağ et al. [78] did identify not only challenges for Scrum Masters but also found seven challenges for Agile Coaches. These challenges include:

- Dealing with incorrect practices of agile development
- Dealing with doubts in people about changes
- Establishing a culture of continuous improvement
- Dealing with black and white mindsets
- Dealing with closed-mindedness
- Establishing a common understanding of agile thinking and practices
- Building an effective coaching model

In contrast to the challenges of Scrum Master, which mostly arise exclusively in the large-scale context, only one of the challenges of Agile Coaches is new in scaled agile approaches. While all of the above items were already known in small, co-located teams, 'Establishing a common understanding of agile thinking and practices' has only occurred when scaling agile methods [78].

2.4. Patterns

2.4.1. Terminology

Most publications [14, 20] about patterns refer to Christopher Alexander's definition of patterns: *Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.* [3] The three elements **context**, **problem** and **solution** occur in every definition found, for example in Eloranta et al. [30], where a pattern is 'a general reusable solution to a commonly occurring [*sic*] problem.' [30]. Elssamadisy [31] characterizes a pattern as 'a particular problem and its solution context' [31]. Patterns are not made up or created by domain experts, but are identified in practice and expressed in written form [31]. According to Buschmann et al. [14], patterns help to document 'existing, well-proven experience' and to endorse good practice. Coplien [18] defines what this thesis references to as the *Rule of Three:* '[...] a good pattern should have three examples that show three

insightfully different implementations.'. A solution to a problem has to be applied at least three times in order to be called a pattern. This rule enables validation of patterns and makes them more trustworthy [31]. Ernst [32] combines all definitions from above and combine them to a very general definition, which this thesis will also use: *A pattern is a general, reusable solution to a common problem in a given context.*
textbfAnti-Patterns are very similar to patterns, as they also describe recurring solutions to a problem; however, these solutions lead to negative consequences. They additionally describe a *refactored solution*, which explains what to do instead [12]. Corresponding to the validity of patterns, anti-patterns also need to occur at least three times in order to become an anti-pattern [32].
Several patterns can also be combined and are allowed to cross-reference each other. The result can either be a **Pattern Language**, or a **Pattern Catalog**. A pattern language is a collection of patterns, organized in a way that the user of such a language is able to find a solution to his 'artifact' [18, 30]. Eloranta et al. [30] propose two approaches for creating the structure of a pattern language. First, the problem-centered approach defines several larger patterns and places minor patterns in a top-down way below the larger ones. For example, Alexander [2] first describes patterns for cities and eventually patterns for rooms. The second approach is the solution-centered approach, in which the pattern language is structured based on the relationships between the solutions. An example provided by Eloranta et al. [30] is the order in which patterns need to be applied, if patterns require other patterns to be applied before. Coplien and Harrison [20] defines a pattern language as 'a language that comprises patterns and the rules to put patterns together in meaningful ways, in a certain sequence.'. In contrast to a pattern language, a pattern catalog classifies each pattern into a category, not requiring them to be connected [32]. Additionally, a pattern language aims at completeness, which is not the case for a pattern catalog [32]. Therefore, this thesis develops the structure for a pattern language, aiming for completeness at a long-term view. However, the first patterns presented can only be considered a pattern catalog due to the lack of completeness.

2.4.2. Pattern Documentation

Patterns are documented according to a predefined schema, consisting of the different sections a pattern has. Three sections form the basis of every pattern [3, 14] are the context, problem, and the solution section. The context section describes the circumstances in which a problem occurs [14] and may indicate if other patterns have already been applied [20]. Based on the context, the user of the pattern language can decide, if the pattern fits his situation of not [20]. The problem section describes the recurring problem within the context [14]. This section can also include several forces, that 'discuss the problem from various viewpoints' [14] and provide more details to the problem. Finally, the solution section explains how to solve the problem in the given context, taking the forces into account [14]. There are several standardized formats for documenting patterns, such as the Portland Form [22], the Alexandrian Form [2] or the Coplien Form [20]. However, in gen-

eral, each author decided on his own schema, depending on the intention of the patterns. Therefore, section 3.2 will describe related pattern language and analyze the way their patterns are documented.

2.4.3. Best Practices for Pattern Writing

There are some aspects that distinct good patterns from other patterns. Meszaros and Doble [59] mention the following mandatory sections that need to be included: Name, Context, Problem, Forces, Solution. Wellhausen and Fiesser [82] also argue positive and negative 'Consequences' as mandatory for every pattern. The problem should be free of context, enabling other patterns to refer to the same problem. The solution should resolve some or all forces, but can also ignore all forces. Meszaros and Doble [59] argue that 'the most appropriate solution to a problem [...] is the one that best resolves the highest priority forces'. Other sections can be included in the pattern if necessary, for example 'Indications', 'Resulting Context', 'Related Patterns', 'Examples', 'Aliases' or 'Rationales' [59]. Best practices for writing patterns includes not only a good schema but also an easy writing style. The reader should be able to clearly distinguish between the sections and quickly identify which sections are important to him and which not. By placing the context section at the top of the pattern, the reader can easily see if the pattern is suitable for him or not [59]. Further, references to other patterns within the text should have different typesetting than the rest of the pattern description [59]. Further, the name of the pattern should summarize the solution section very short [59] by using a 'noun phrase' [15]. On the other hand, a pattern could also be named after the artifact it creates [59]. Further, Meszaros and Doble [59] propose to include a 'Pattern Intent' on top of each pattern, which summarizes in one to two sentences what the pattern is about. The terminology used should be understandable for the target audience [59]. Another way to provide structure to a pattern language is to include problem-solution tuple at the end of the document describing the patterns. In [20] this is called 'Patlets'. Eventually, Harrison [39] points out to be aware of 'weasel words', that do not have any meaning but sound good. Patterns with weasel words look good on paper, but can not be applied in practice. Harrison [39] mentions the example of 'Carefully design the system.', 'carefully' being the weasel word. He also highlights not to write the problem by rephrasing the solution section in the following way:
Problem: You are not doing X.
Solution: Do X.
While Meszaros and Doble [59] focus on structuring a pattern to provide as much value to the reader as possible, Wellhausen and Fiesser [82] explain the process of writing a pattern from the author's perspective. They propose first to write the solution and then the problem. Afterward, the consequences should be evaluated, and the forces should be written. The context section should be written at the very end. This process is shown in Figure 2.4 and is also described by Harrison [39], however omitting the consequences. The authors further point out to divide the consequences into two parts: first, the benefit part and second, the liabilities part. The benefits describe what positive consequences the appli-

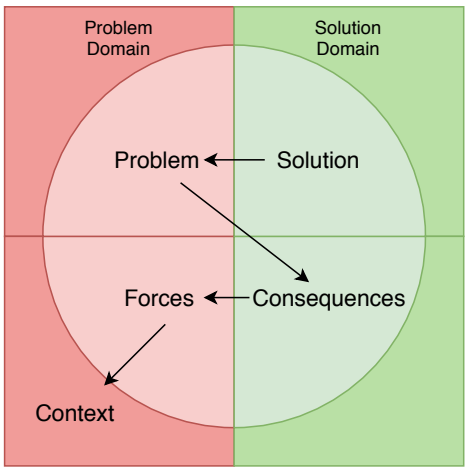


Figure 2.4.: The Process of Pattern Writing based on Wellhausen and Fiesser [82]

cation of the solution brings up, whereas the liabilities sum up all drawbacks and things the reader should be aware of when applying the solution [82]. The forces section details the problem and explains why it is hard to solve the problem [82]. Every force should be dismissed by consequence, resulting in all forces being absent after applying the solution [82]. Finally, the context should be the same before and after the solution, consequently must not change [82].

3. Related Work

This chapter summarizes existing research in the fields of Large-Scale Agile Development and Pattern Languages intending to justify the demand for a pattern language in this domain. For this purpose, the first part of this chapter describes relevant research conducted aiming at identifying challenges in Large-Scale Agile Development (see Section 3.1). The second part explains the characteristics of pattern languages containing organizational, project management, and agile patterns and compares them to the approach of this thesis (Section 3.2).

3.1. Related Work on Challenges in Large-Scale Agile Development

Uludağ, Ö. (2018) - Identifying and Structuring Challenges in Large-Scale Agile Development based on a Structured Literature Review

As part of the same research project, the publication by Uludağ et al. [78] serves as the starting point of this work. Based on a structured literature review including 73 relevant sources, they identified fourteen stakeholders in large-scale agile projects and 79 challenges within eleven categories. The identified stakeholders include Development Team, Product Owner, Scrum Master, Software Architect, Test Team, Product Manager, Program Manager, Agile Coach, Enterprise Architect, Business Analyst, Solution Architect, Portfolio Manager, Support Engineer, and UX Expert. The challenges are split into the categories Culture & Mindset, Communication & Coordination, Enterprise Architecture, Geographical Distribution, Knowledge Management, Methodology, Project Management, Quality Assurance, Requirements Engineering, Software Architecture, and Tooling. Based on the approach proposed by Cruzes and Dyba [21], they built a list of six codes inspired by Ernst [32], including stakeholders, challenges, methodology patterns, architecture principles, viewpoint patterns, and anti-patterns. The final artifact represents a mapping from all categorized challenges to the stakeholders as well as an indication if a challenge raised newly in the large-scale context, compared to small, co-located teams. However, the paper only presents challenges and stakeholders identified in the literature, missing validation of the results in practice. Further, not all challenges are documented in the literature, consequently increasing the demand for a practice-oriented analysis.

Dikert et al. (2016) - Challenges and success factors for large-scale agile transformations: A systematic literature review

In contrast to Uludağ et al. [78], Dikert et al. [26] not only identify 35 challenges, classified into nine categories but also describe 29 success factors, classified into eleven categories, based on 52 papers. Even though this paper served as input for [78], the identification of success factors differentiates the two publications and consequently gaining relevance for this work. Because 90% of the literature included has been depicted as 'experience reports' [26], its practical relevance is high. Table 3.1 summarizes all identified categories challenges and success factors.

Categories of Challenges	Categories of Success Factors
Change resistance	Management Support
Lack of investment	Commitment to change
Agile difficult to implement	Leadership
Coordination challenges in multi-team environment	Choosing and customizing the agile approach
Different approaches emerge in a multi-team environment	Piloting
Hierarchical management and organizational boundaries	Training and coaching
Requirements engineering challenges	Engaging people
Quality assurance challenges	Communication and transparency
Integrating non-development functions	Mindset and Alignment
	Team autonomy
	Requirements management

Table 3.1.: Categories of Challenges and Success Factors in Large-Scale Agile Development identified by [26]

3.2. Related Work on Pattern Languages

Several authors have faced the challenge to identify patterns in the software development domain. The following sections discuss different pattern languages addressing both organizational and agile patterns within the software development domain.

Coplien (1995) - A Generative Development-Process Pattern Language

The earliest publication by Coplien [17] presents a collection of patterns for shaping an organization and its process of developing software [17]. Overall, it includes 42 patterns, distributed into two categories: Process Patterns and Organizational Patterns. It does

not focus exclusively on large organizations; however the pattern *Pattern 9: Organization Follows Location* presents a solution to the problem of distributed teams. Nevertheless, it misses the context of agility as the patterns describe detailed schedules and the duration of a project. In contrast, some patterns, like *Pattern 2: Self-Selecting Team* or *Pattern 12: Patron*, already contain agile elements, without calling it an agile practice. According to Coplien [17], a Patron is a manager, who removes impediments and takes care of the spirit of the organization. These tasks are very similar to the responsibilities of a Scrum Master. All patterns follow the same schema, including a Name and an optional Alias, followed by the Problem and Context section. The problem is more detailed by a Forces section, and finally, a short Solution and the Resulting Context is explained. In the end, a Rationale section describes how the pattern has been identified and how it can be combined with other patterns of this pattern collection.

Harrison (1996) - Organizational Patterns for Teams

Harrison [38] identifies four patterns for teams to work together to design and develop software. The introduction is written as a pattern itself, defining the sections problem, context and forces commonly for all four patterns, whereas the solution section contains references to each pattern. Each of the four patterns individually defines its own context, problem, forces and solution section, which add onto the introductory ones. Additionally, each pattern describes its rationale and possible variation as well as a set of related patterns from other publications. The names of the four patterns are: *Unity of Purpose*, *Diversity of Membership*, *Lock 'em Up Together*, and *Validation by Teams*. The pattern collection focuses on how to get several people to work towards a common goal, how to get the right resources, how to create a consistent software architecture, and how to validate this design. However, these patterns are not connected to each other and do not form a pattern language. Furthermore, Harrison [38] does not categorize them, making them just a small collection of patterns on the same topic. Agility is also not addressed by these patterns.

Taylor (2000) - Capable, Productive, and Satisfied: Some Organizational Patterns for Protecting Productive People

In contrast to Harrison [38], Taylor [75] has created a pattern catalogue containing nine patterns, divided into four categories. These patterns intend to create, maintain and preserve a 'productive team culture' [75] for software development teams. The theoretical foundations for these patterns are the author's practical experience of twelve years in software development. The root pattern for all patterns is called *Production Potential*, which describes a way to make teams productive. Taylor [75] argues, that most teams work a lot, but produce a little outcome. The other patterns include steps that support the fulfillment of the root pattern, either by establishing, maintaining or preserving the production potential. Each pattern starts with a sentence summarizing the intention of it, explains the context afterward and continues with a description of the problem. Next, different

forces are elaborated and emphasized by a picture. The solution section describes how the problem can be solved within the context, always including the *therefore*-phrase. Each pattern concludes with a description of the resulting context, known problems, and related patterns within the pattern language. Similar to Harrison [38] this pattern language also does not explicitly refer to agile teams. Further, the size of the teams addressed by these patterns is not specified.

Coplien and Harrison (2004) - Organizational Patterns for Agile Software Development

While the three pattern languages mentioned above do not focus on agile teams, the following languages were specifically developed for the agile software development domain. The most well-known publication was done by Coplien and Harrison [20], who created four different pattern languages, encompassing 94 patterns: Project Management Pattern Language, Piecemeal Growth Pattern Language, Organizational Style Pattern Language, and People and Code Pattern Language. The first two languages are referred to as *Design Pattern Languages*, which 'lay the foundation of the entity to be built' [20], while the last two are called *Construction Pattern Languages*, which 'deal with [...] creating things' [20]. Coplien and Harrison [20] explain the patterns within each pattern language build upon each other. Further, patterns reference each other beyond the borders of a distinct language. Besides, several patterns occur in several languages (e.g., UNITY OF PURPOSE). The patterns are written according to the Alexandrian format [20], including an image and a short story explaining the image, a description of the context, a problem summary as well as forces and a solution section, which contains the keyword *therefore*. Each pattern is concluded with a discussion section describing related patterns. The context and the problem sections, as well as the solution and the discussion section, are each divided by three diamond symbols. Further, each pattern is classified with a confidence level, expressed by up to two stars next to the name. The confidence level illustrates how often a pattern has been observed in practice and how confident the authors are about a pattern. The patterns were identified using interviews, social network techniques to build organizational models and finally, agile teams evaluated the captured patterns. Eventually, the patterns have been assigned to the different pattern languages, and meaningful links between patterns have been identified.

Elssamadisy (2008) - Agile Adoption Patterns: A Roadmap to Organizational Success

Another example of a collection of agile patterns has been published by Elssamadisy [31], who developed a pattern catalog for the adoption of agile practices. The patterns describe problematic situations and refer to an agile practice to fix it. Overall, there are 38 patterns divided into four categories: feedback practices, technical practices, supporting practices, and clusters. The pattern schema differs from the other pattern languages by a higher number of sections. While each pattern still has the standard sections description, context and forces, this pattern language also documents a dependency diagram, showing

'interpractice *[sic]* dependencies (for practices) and groupings (for clusters)' [31]. Further, s sketch section includes a fictional story, illustrating the problem and applied solution. The characters for this section have been introduced in a separate chapter and re-appear throughout the book. The forces section is followed by a 'Therefore' section, explaining the solution. Also, an 'Adoption' section defines concrete steps to adopt the solution explained before. Negative consequences are revealed in a 'But' section, followed by a list of variations of the solution. Another section that has not been observed in other pattern language is called 'Business Value', which is 'a short description of the business values that this practice or cluster improves' [31]. Each pattern is concluded with a list of further external references one can read more about the pattern.

Välimäki (2011) - Pattern Language for Project Management in Global Software Development

A more precise scope is provided by Välimäki [79], who defines patterns for increasing performance of global project management. He emphasizes that there are practices applying on both, co-located and distributed projects; however, his pattern language concentrates on distributed projects only. The language contains 18 patterns, which are categorized across the PRINCE2 process steps: *Directing a Project*, *Starting up a Project*, *Initiating a Project*, *Controlling a Stage*, *Managing Stage Boundaries*, *Managing Product Delivery*, *Planning* and *Closing a Project*. Similar to Coplien and Harrison [20], a pattern can fit into several categories. The schema used for documentation is a subset of the Appleton format, including the following sections: initial context, roles, forces, solution, resulting context.

Beedle et al. (1999) - SCRUM: A Pattern Language for Hyperproductive Software Development

Finally, several authors documented well-known Scrum practices as patterns [7, 9]. The earliest publication by Beedle et al. [9] describes three common Scrum practices in more detail: Sprint, Backlog, and SCRUM Meetings. The patterns are written in a way that the three Scrum practices are the solution to a problem, while the context does not describe an already agile project. Therefore, this pattern languages provides agile solutions to non-agile problems for small, co-located teams. The patterns are documented by a description of the context first, followed by the problem and the forces. In the pattern *SCRUM Meetings*, the forces section is yet divided into 'Estimation', 'Planning' and 'Tracking'. After the forces, the solution and the rationale are discussed and concluded by a summary of known uses of this solution. Finally, the resulting context is shortly explained, which includes most general benefits of agile methods. In a later extension of the pattern language [8], the context section is enriched with Scrum-dependent descriptions and the three categories in the forces section are explicitly called 'Misfit Variables'. They are defined as 'variables in the context that can adjusted *[sic]* to fit the solution to solve the problem' [8]. Additionally, this extension contains a large number of references to the patterns by Coplien [17].

Beedle et al. (2010) - Essential Scrum Patterns

Beedle et al. [7] presented another, updated collection of Scrum patterns ten years later. However, the paper is not finished and lacks content at some points. Scrum best practices included in pattern form are Product Backlog, Product Owner, Scrum Team, Scrum Master, Sprint Backlog, Sprint Planning Meeting, Sprint, Daily Scrum, Sprint Burndown, Sprint Review, and Sprint Retrospective. The first four patterns do not explicitly have distinct sections; however, the structure indicates the following schema: Context, ***, Problem, ***, Forces, Solution, Resulting Context. The other patterns contain the names of the sections in front of them. They cover the same schema, however sometimes dismissing the forces and obliterating the delimiting stars. No complete version of this paper has been found.

ScrumPLoP Website

The website of the ScrumPLoP conference [71] collects all patterns presented at their conference. It contains 234 patterns; however, some of them are deprecated and not accessible anymore. The patterns are divided into nine categories: Value Stream, Team, Sprint, Process Improvement, Product Organization, Distributed Scrum, Scaling Scrum, Scrum Code, and Misc [71]. Remarkable authors contributed to this collection, such as James Coplien, Neil Harrison, Mike Beedle or Jeff Sutherland. The schema of the patterns differs from pattern to pattern as there are no concrete guidelines or rules. However, every pattern contains to most essential sections: a picture, context, forces, 'therefore', examples, related patterns, and references. Many of the patterns have delimiting diamonds after the context and after the solution. The scope of this pattern collection is to document agile best practices; however, finding patterns suitable for scaled agile is difficult as there is no guiding structure.

Mitchell (2016) - Agile Development in Practice

Mitchell [60] presents 49 patterns in the Scrum context, divided into four categories: Agile Patterns of Method, Agile Patterns of Responsibility, Agile Patterns of Representation and Anti-Patterns. Those patterns are only described in the appendix of the publication and do not build the main artifact. Even though there is no official definition of the meaning of each category, their content leads to the following interpretation: 'Agile Patterns of Method' contains patterns describing a certain methodology, for example, *Inspect and Adapt*, *Iterate* or *Scrumban*, while 'Agile Patterns for Responsibility' handles role concepts, like *Servant Leadership* or *Product Ownership*. Eventually, 'Agile Patterns for Representations' describe artifacts and visualizations, for example *Avatar*, *Epic* or *Information Radiator*. The schema is consistent throughout all patterns, starting with a name and the description of the intent. The section 'Proverbs' contains short sentences that seem fitting to the author. For example, the proverb to the pattern *Metrics* is 'You can't manage what you don't measure' [60]. The proverbs are followed by a list of aliases for the pattern and a descrip-

tion of the motivation. The 'Structure' section summarizes the solution, which is extended by an 'Applicability' section. Benefits and liabilities are explained in the 'Consequences' section, and finally, real-life examples are provided in the 'Implementation' section. Each pattern also contains a mixture of a Unified Modeling Language (UML) class and a UML Use Case diagram to illustrate the components of the pattern.

Summary

The presented pattern languages all have their benefits and drawbacks. The biggest deficiency they all have in common is the absence of a profound structure that guides the user through the patterns. This missing link is provided by the LSADPL presented in the next chapter. By adding structural elements to the language, the user can see which patterns are relevant to him, dismissing all other. Hence, a Product Owner can directly view all patterns relevant to him and must not inspect patterns addressing the concerns of developers. Further, none of the pattern languages solely focuses on Large-Scale Agile Development, making it hard for stakeholders to identify suitable patterns.

Source	Scope & Goal	Focus on Agile	Number of Patterns	Categories
Coplien [17]	Collection of patterns for shaping a new organization and its development process	Partially	42	(1) Process Patterns (2) Organizational Patterns
Harrison [38]	Collection of patterns for creating effective software development teams	No	4	-
Beedle et al. [9]	Collection of Scrum patterns	Yes	3	-
Taylor [75]	Collection of patterns for creating product software development environments	No	9	(1) Establishing a Production Potential (2) Maintaining a Production Potential (3) Preserving a Production Potential
Coplien and Harrison [20]	Collection of organizational patterns that are combined into a set of four pattern languages	Yes	94	(1) Project Management (2) Piecemeal Growth (3) Organizational Style (4) People and Code
Elssamadisy [31]	Collection of patterns for successfully adopting agile practices	Yes	38	(1) Feedback Practices (2) Technical Practices (3) Supporting Practices (4) The Clusters
Beedle et al. [7]	Collection of the most essential best practices of Scrum	Yes	11	-
Välimäki [79]	Enhancing performance of project management work through improved global software project management practice	Partially	18	(1) Directing a Project (2) Starting up a Project (3) Initiating a Project (4) Controlling a Stage (5) Managing Stage Boundaries (6) Closing a Project (7) Managing Product Delivery (8) Closing

Source	Scope & Goal	Focus on Agile	Number of Patterns	Categories
Mitchell [60]	Collection of patterns to address agile transformation problems	Yes	49	(1) Patterns of Method (2) Patterns of Responsibility (3) Patterns of Representation (4)Anti-Patterns
ScrumPLoP [71]	Body of pattern literature around agile and Scrum communities	Yes	234	(1) Value Stream (2) Team (3) Sprint (4) Process Improvement (5) Product Organization (6) Distributed Scrum (7) Scaling Scrum (8) Scrum Core (9) Misc

Table 3.2.: Comparison of Pattern Languages focusing on Organizational and Agile Patterns

Coplien [17]	**Harrison [38]**	**Beedle et al. [9]**	**Taylor [75]**	**Coplien and Harrison [20]**
Name, Alias, Problem, Context, Forces, Solution, Resulting Context, Rationale	Name, Problem, Forces, Solution, Rationale, Variation, Related Patterns	Name, Context, Misfit Variables, Problem, Forces, Solution, Examples, Resulting Context, Rationale	Name, Intent, Context, Problem, Forces, Solution, Resulting Context, Known Problems, Related Patterns	Name, Image, Short Story, Context, *** Problem Summary, Forces, Therefore, Solution, *** Discussion, Related Patterns, Confidence Level
Elssamadisy [31]	**Beedle et al. [7]**	**Välimäki [79]**	**Mitchell [60]**	**ScrumPLoP [71]**
Name, Description, Dependency Diagram, Sketch, Context, Forces, Therefore, Adoption, Variations, But, Business Value, References	Name, *** Context, *** Problem, Forces, Solution, Resulting Context	Identifier, Name, Initial Context, Roles, Forces, Solution, Resulting Context	Name, Intent, Proverbs, Also Known As, UML Diagram, Motivation, Structure, Applicability, Consequences, Implementation	Irregular

Table 3.3.: Comparison of Related Pattern Languages' Sections

4. Evolving the Large-Scale Agile Development Pattern Language

The last chapter summarized related pattern languages from the software development domain. Whereas some focused on agile practices [7, 8, 9, 31] and others included patterns for distributed development teams [17, 79], none of the presented pattern languages solely targets problems in the field of large-scale agile development. Therefore, the following chapter will describe the entire development process of the LSADPL. The language has been developed in an incremental order, which is also displayed by this chapter's structure. Section 4.1 will describe the development of the first version and explain how this version has been evaluated by fourteen expert interviews. The results of this evaluation (see Section 4.2) delivered the input for the second version of the LSADPL, which is described afterwards in Section 4.3.

4.1. First Version

4.1.1. Description

The UML diagram in Figure 4.1 represents the first draft of the LSADPL. It consists of eight types of elements: *Stakeholder*, *Initiative*, *Challenge*, *Principle*, *Anti-Pattern*, *Coordination Pattern*, *Viewpoint Pattern*, and *Methodology Pattern*. The ninth element LSAD Pattern is an abstract class, which encapsulates the classes Coordination Pattern, Methodology Pattern, and Viewpoint Pattern. The attributes of the classes represent the different sections of the pattern types and concepts. The elements build on the findings in [78], where a literature research has been applied to identify challenges and stakeholders in the large-scale agile development domain. Uludağ et al. [78] developed a list of codes inspired by Ernst [32] including Stakeholder, Challenges, Methodology Patterns, Architecture Principles, Viewpoint Patterns, and Anti-Patterns. However, these categories cannot be mapped completely onto the LSADPL as they only focus on the Enterprise Architecture Management (EAM) perspective. As described in Section 2.1, principles and values are the foundation of agile software development. Therefore, the LSADPL needs a general *Principle* category, including relevant principles for all stakeholders and not only architecture-relevant principles. Furthermore, a challenge can be possessed not only by a single stakeholder but by a whole team or even a whole enterprise, consisting of many different stakeholders. For this reason, the LSADPL introduces the concept of *Initiatives*. Besides, the aforementioned *Coordination Pattern* has been added to the pattern language, because many frameworks

propose mandatory events as an outline, but no concrete methods to apply during events. Further, the coordination of multiple agile teams is a key challenge in large-scale agile development [27]. As a consequence, we explicitly differentiate between coordination mechanisms and methodologies.

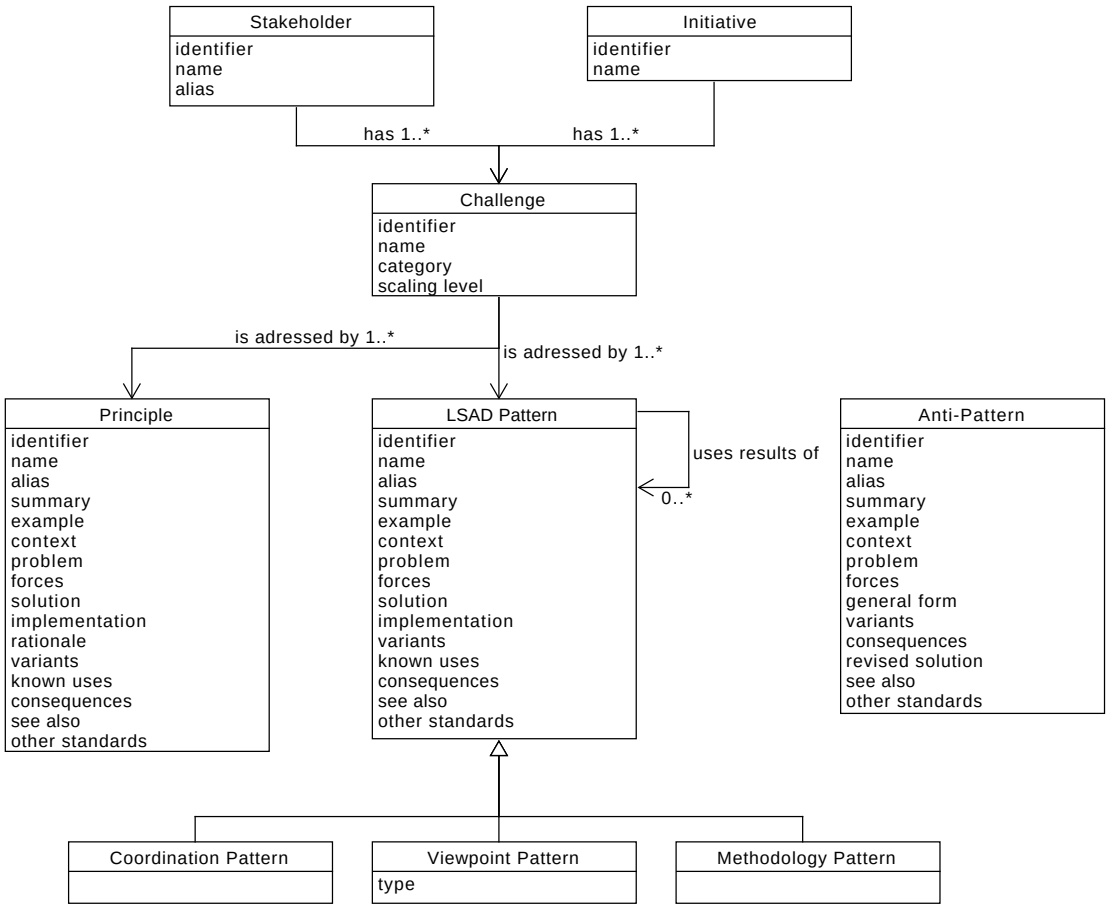


Figure 4.1.: Initial Version of the Large-Scale Agile Development Pattern Language

4.1.2. Elements

The elements of the pattern language are defined as follows:
Stakeholder: Stakeholders are all people, who are actively involved in, have an interest in, or are in some way affected by large-scale agile development [78]. Each stakeholder has a unique identifier, a name, and can have multiple aliases. Besides, each stakeholder

references to a set of challenges. Possible stakeholders of large-scale agile development have been identified in [78]: UX Expert, Agile Coach, Solution Architect, Test Team, Software Architect, Scrum Master, Enterprise Architect, Portfolio Manager, Program Manager, Business Analyst, Product Manager, Support Engineer, Development Team, and Product Owner.

Initiative: This element is supplementary to the element *Stakeholder*, as it can be used if a group of stakeholders on different scaling levels is affected by a challenge. An initiative can be one of the following: Team Initiative, Program Initiative, Portfolio Initiative, IT Organization Initiative, Enterprise Initiative. I.e., in a team initiative, the whole team is affected by a challenge, not only the Scrum Master. Similar to stakeholders, each initiative has a unique identifier as well as a name and a set of challenges.

Challenge: A challenge is everything that hinders the stakeholders in performing tasks in the large-scale agile development domain. Each challenge is categorized into one of the following classes: Culture & Mindset, Communication & Coordination, Enterprise Architecture, Geographical Distribution, Knowledge Management, Methodology, Project Management, Quality Assurance, Requirements Engineering, Software Architecture or Tooling [78]. Furthermore, a challenge includes the attribute *Scaling Level*, which is similar to the aforementioned concept of *Initiative*. Challenges refer to principles and patterns that help to solve the challenge.

Principle: Principles are general rules and guidelines that stakeholders should comply with. A principle can also be measured by a Key Performance Indicator (KPI), which is described in a *Viewpoint Pattern*. This category includes not only architecture principles (e.g., Loose coupling [84]) but also agile and lean principles defined in frameworks or any kind of principles that are specified by teams or organizations for better collaboration.

Coordination Pattern: A coordination pattern's solution consists of a coordination mechanism, which can either be digitally or in person. It solves at least one challenge and refer to a *Methodology Pattern* or a *Viewpoint Pattern*. For example, during the coordination meeting Sprint Retrospective the viewpoint Starfish-Board can be used.

Methodology Pattern: A methodology pattern's solution describes concrete processes, practices or methods. A methodology pattern can refer to *Viewpoint Patterns*. For instance, the prioritization of tasks is defined differently based on the context (Methodology Pattern) and can be complemented by KPIs, which are documented as Viewpoint Patterns.

Viewpoint Pattern: A viewpoint pattern defines ways to visualize information in the form of documents, boards, metrics, models, or reports. This pattern class can be used by other patterns or can directly address a challenge. Metrics can further be used to measure compliance with a principle.

Anti-Pattern: An Anti-Pattern is a common practice that has negative consequences while providing a revised solution. An Anti-Pattern does not address a certain stakeholder, initiative, or challenges. It has no connections to other elements within the pattern language.

4.1.3. Sections

The sections of the elements LSAD Pattern, Principles, and Anti-Pattern have been selected based on the findings in Subsection 2.4.2 and Section 3.2. We do not use a specific format; however, we included the three most important sections 'Context', 'Problem' and 'Solution' according to the definition of a pattern [3]. The sections 'Identifier', 'Name', 'Alias' and 'Summary' have been taken from [32] and serve as a short outline at the top of each pattern, anti-pattern or principle. The section 'Example' has been taken into the schema, because it increases the reader's understanding. 'Forces' is important as in our pattern language, the problem-section only refers to challenge items. Therefore, the 'Forces' section describes the problem in more detail and what the real pain points are. To enable higher usability of the pattern language, we distinguish between the 'Solution' and the 'Implementation' section and extend these with a 'Variants' section. This allows the user to understand how to implement a rather abstract solution directly. The counterpart to the 'Forces' section is presented in the 'Consequences' section. In the best case, the benefits in the 'Consequences' section override all forces with as few liabilities as possible. The 'Known Uses' section acts as validation for the user that the solution presented has been successfully applied in several other companies. As a Viewpoint Pattern includes several artifacts of different types, the section 'Type' has been added to the element *Viewpoint Pattern* only. The section 'Rationale' can also only be found in the element *Principle* and the sections 'General Form' and 'Revised Solution' only apply to the element *Anti-Pattern*. The sections shown in Figure 4.1 are defined in Table 4.1.

4.2. Evaluation

4.2.1. Methodology

The scope of the evaluation was to examine whether the LSADPL is valuable, logical, and understandable to the user. The user is any stakeholder who is involved in or affected by large-scale agile development. The focus was to collect feedback regarding the following questions:

1. Is the LSADPL relevant to the industry?
2. Do the elements of the LSADPL provide value to the user?
3. Is the structure of the LSADPL logical and understandable?
4. Is the LSADPL complete?

To answer these questions, we have set up a series of interviews that we called *evaluation interviews*. The scope of these interviews laid on examining the relevance of each element of the language, as well as on the connections between each pair of elements. Specific parts of the language that had raised further discussions during the development

Section	Description
Identifier	A unique ID for easier identification and cross-referencing
Name	A unique, concise name summarizing the solution
Alias	How this pattern is also known as
Summary	Three to five sentences summarizing the context and solution
Example	A fictional story, in which the solution has been applied, can also include a description of the context and forces
Context	Circumstances under which the solution can be applied to the problem
Problem	References to one or more *Concern* entities
Forces	Describes why the problem is hard to solve
Solution	The solution to the problem within the context
Implementation	How to adapt the solution within an organization or team
Variants	Lists possible variants of the same solutions
Known Uses	List of organizations in which this pattern has been applied
Consequences	Positive consequences (benefits) and negative consequences (liabilities) when applying the solution
See Also	References to other LSAD Patterns or Principles
Other Standards	References to other standards which uses this pattern
Type	Categorizes the Viewpoint Pattern, e.g. Board, Blog, KPI (*only applicable to Viewpoint Patterns*)
Rationale	Explains why a principle is necessary and what its intention is (*only applicable to Principles*)
General Form	Problematic solution to the problem (*only applicable to Anti-Patterns*)
Revised Solution	How the problem should be solved instead (*only applicable to Anti-Patterns*)

Table 4.1.: Sections used in the first version of the Pattern Language

of the language have been explicitly evaluated during the interviews. These debatable parts included the inclusion of anti-patterns and their connection to other elements of the LSADPL, as well as the integration of principles into the LSADPL. Further, we wanted to explicitly evaluate, whether a distinction between M-Patterns and CO-Patterns is valuable. We designed a questionnaire to address the aforementioned questions, which can be found in Appendix A.1. The questionnaire consists of four sections: (1) general questions about the interviewee, (2) questions about individual elements of the LSADPL, (3) questions about the relations between all elements, and (4) a discussion section.

General Questions. This introductory section collects information about the respondent, his experience with large-scale agile development and general information about his background. In a preliminary version of the questionnaire, we only include the following ques-

tions:

1. How long have you been working in your company?
2. How is your role denoted in your company?
3. How long have you been dealing with the topic of large scale agile development?
4. How long has your company been conducting large scale agile development?

All these questions were initially open questions, so not answer options were provided. However, we removed the first question as this is not relevant for our evaluation and would not influence our results. The second question was moved to the top of the questionnaire, making it the first question to be asked. The role is crucial for us to know so we can interpret which elements are more relevant to which stakeholder. To get an accurate role description, the respondents were not restricted to a set of answer options but could answer freely. We slightly altered the third question and asked for experience in large-scale agile development, while providing four possible answer options: less than one year, one to three years, three to six years or more than six years. This allowed for better and more precise evaluation results. The last question stayed the same but was also extended with the same four answer options.
the development of the questionnaire, we added four more questions to this section:

1. Is your company acting on national or international level?
   Answer Options: National, International
2. In which country is your company's headquarter located?
   Answer Options: the interviewees could name any country
3. Which sector does your company belong to?
   Answer Options: Agriculture/Mining; Building Industry; Education/University; Finance/Insurance/Real Estate; Government; Health Care; IT/Technology; Production; Retail; Service Industry; Transportation/Logistics; Communication/Supply; Non-Profit; Other
4. How many people does your company employ?
   Answer Options: 1 - 10; 11 - 50; 51 - 250; 251 - 500; 501 - 1.000; 1.001 - 2.000; 2.001 - 5.000; 5.001 - 10.000; 10.001 - 50.000; 50.001 - 100.000; 100.001 - 200.000; more than 200.000

These questions are necessary to examine possible dependencies between the size and sector of a respondent's company and his answers.
Questions regarding elements of the LSADPL. The second block of the questionnaire contained a set of questions for each element of the pattern language. We asked the following two questions for each of the element:

1. In your opinion, how valuable is the inclusion of the element <*element*> into the concept model of the LSADPL?

2. Why would the element <*element*> be valuable/not valuable to you?

The first question could be answered on a five-point Likert scale from *very valuable* (1) to *not valuable at all* (5). This enables descriptive statistical analyses and facilitates comparability of the interviews. By asking for a justification in the second question we got direct feedback, which helped us in enhancing the LSADPL. As initially stated, we got several debatable points within the pattern language that we needed an assessment of. Therefore, we explicitly asked, if a distinction between *Coordination Pattern* and *Methodology Pattern* is helpful (answer options were 'yes' or 'no') and why it is (not) helpful. Further, we asked if the element *Anti-Pattern* should be connected to other elements, and why.
Questions regarding the relations within the LSADPL The third section aimed at evaluation of the connections between the individual elements. Therefore, we again asked on a five-point Likert scale from *(1) Very meaningful* to *(5) Not meaningful at all* how meaningful the relation between two elements is and why. This way we could examine if the reference within one item onto another item is relevant or not.
Discussion. The last block invited the respondent to talk about concepts that he wishes to see included in the pattern language, that are not integrated yet, as well as asked for general feedback regarding the LSADPL and the interview concept. Finally, we had two concluding questions that both could be answered on a five-point Likert scale from *(1) Very Certain* to *(5) Certainly Not*:

1. Would you use patterns for addressing recurring challenges in large-scale agile development?

2. Would patterns regarding large-scale agile development help you in your job?

These enable a statistical measurement of the usefulness of the LSADPL.
Interview Execution. In a first step, we defined the overall outline of the interviews. We aimed at an overall time frame of 30 to 45 minutes; nevertheless, the first two interviews revealed that we needed to extend the time frame to 60 minutes maximum. As we asked for justification of each answer, interviewees tended to elaborate their justification providing us valuable insights. The interviews were mostly conducted remotely via video conferencing tools. We showed an interactive prototype[1] of the LSADPL to explain the concept and practical use. A screenshot of the prototype is shown in Figure 4.2. Further, we prepared examples for each element, which were explained during the interview. The interviewees were first introduced into the prototype, provided with an example and eventually asked the questions regarding the corresponding element.

[1] `https://scaling-agile-hub.sebis.in.tum.de/#/patterns`

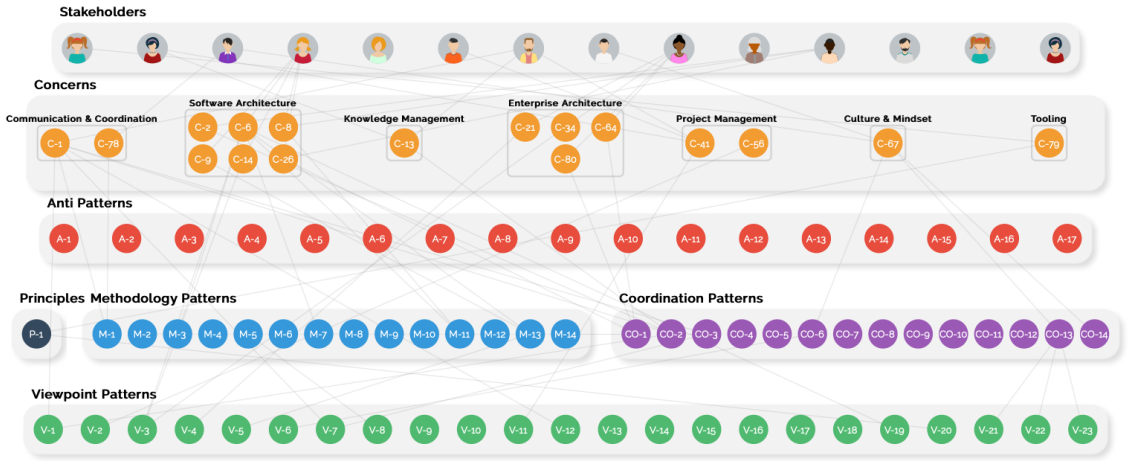


Figure 4.2.: Screenshot of the Large-Scale Agile Development

4.2.2. Results

This sub-chapter summarizes the results of the evaluation interviews. We aimed for an even distribution of stakeholders' roles to get a variety of perspectives. The interviews started in December 2018 and ended in January 2019. Overall, we sent out 22 invitations and conducted 14 interviews, which corresponds a response rate of 63,63%.

Respondents

Table 4.2 shows all interviewees, their role, their own experience in years in working in large-scale agile development as well as how many years their current company pursues large-scale agile endeavours. The majority of respondents (57%) has an experience of three to six years in working in the large-scale agile development domain, while only one respondent has very few experience. 22% declared experience of more than six years. The remaining 14% stated an experience of one to three years. In contrast to the distribution of roles, the distribution of experience was unknown in advance and could not be influenced by picking different respondents. However, the resulting distribution fits the research goals in getting feedback by people with many different backgrounds. The experience with large-scale agile methods within the respondents' companies shows that 57% already apply such methods for three to six years, 36% for one to three years and only 7% for more than six years.

Figure 4.3 shows the distribution of roles, revealing that most respondents have an EAM background. On the other side, Figure 4.4 displays the distribution of the respondent's companies sector, showing that the majority came from the IT sector. Finally, Figure 4.5 represents the different company sizes, revealing a uniform distribution across all kinds of

No.	Alias	Role	Own Experience	Company's Experience
1	EA1	Enterprise Architect	1 - 3	3 - 6
2	PM1	Project Manager	1 - 3	3 - 6
3	DV1	Developer	3 - 6	3 - 6
4	AC1	Agile Coach	3 - 6	1 - 3
5	EA2	Enterprise Architect	3 - 6	1 - 3
6	EA3	Enterprise Architect	3 - 6	3 - 6
7	SA1	Solution Architect	3 - 6	3 - 6
8	PA1	Platform Architect	> 6	3 - 6
9	SA2	Solution Architect	3 - 6	> 6
10	AC2	Agile Coach	3 - 6	3 - 6
11	AC3	Agile Coach	> 6	3 - 6
12	PO1	Product Owner	> 6	1 - 3
13	EA4	Enterprise Architect	3 - 6	1 - 3
14	DV2	Developer	< 1	1 - 3

Table 4.2.: Evaluation Interview Respondents

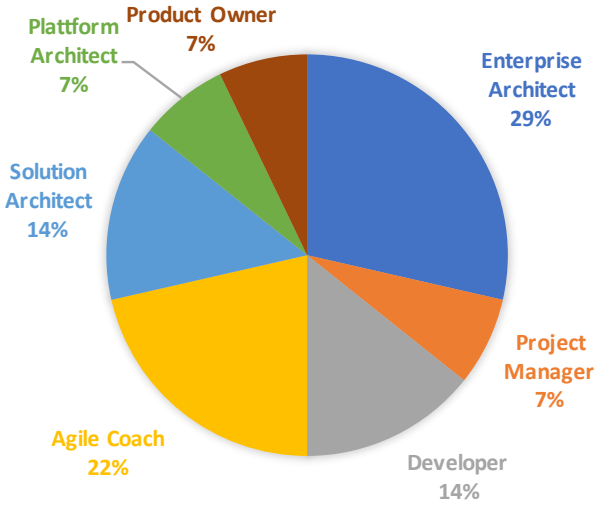


Figure 4.3.: Roles of the Interviewees

enterprises. Companies from the Service Industry sector provide coaching and consulting to other companies. Consequently, participants' own experience with scaled-agile can be very high, although their own company's experience and employee count is very low. Summarized, we gathered evaluation results from people with many different backgrounds, which increases the validity of the evaluation. As the LSADPL should be used by both, experienced and inexperienced people, it is necessary to receive feedback from as different

stakeholders as possible.

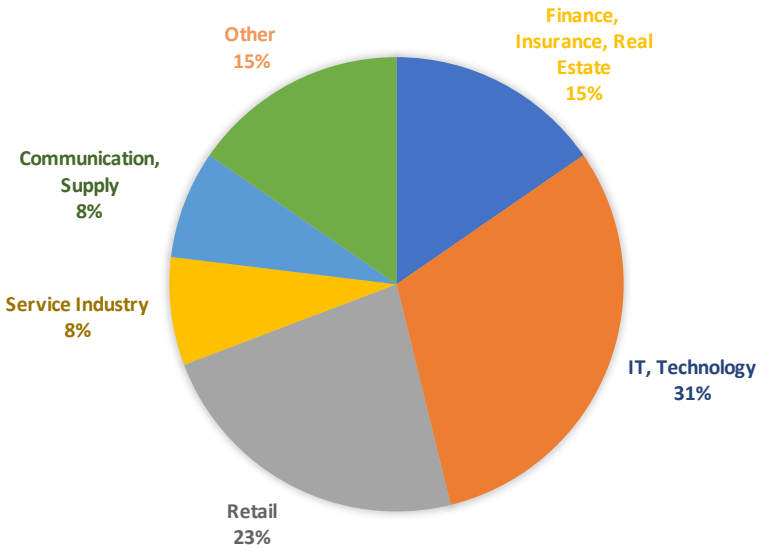


Figure 4.4.: Sectors of the Interviewees' Companies

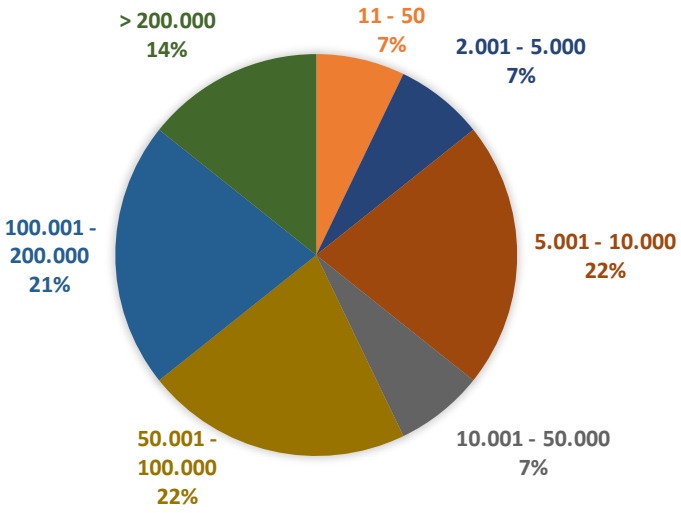


Figure 4.5.: Number of Employees of the Interviewees' Companies

Evaluation of the element Stakeholder

The element *Stakeholder* received an average rating of 1.0 and a standard deviation of 0.0, which is the best possible value. All respondents agreed that the element is *very valuable* for the LSADPL and should not be changed. EA1 stated that it increases transparency as it is clearly visible which stakeholder possesses which challenges. Stakeholders are 'inevitable' (DV1, EA1, EA5) and represent the 'Big Picture' (DV2). PO1 emphasized that the element allows for a better customer-orientation of the pattern language, while AC2 argues that it ensures higher customer satisfaction. However, AC2 and AC3 both disliked the name of the element and would rather call it 'Player' or 'Key User' instead, as they have a different definition of the term 'Stakeholder' in an agile context. Stakeholders in their understanding are any external people having an interest in the agile undertaking, but are not part of the agile team itself. While in the LSADPL Product Owner and Scrum Master are explicitly included as a stakeholder, AC2 and AC3 would not see them as a stakeholder. Still, they admit the demand for including them within the pattern language. As the majority of respondents agreed with the name, we did not make any changes to the element. The connection between *Stakeholder* and *Challenge* also had an average rating of 1.0, a standard deviation of 0.0, and received very positive feedback. AC1 denoted this connection as obvious because problems always lead back to people. Without this connection, stakeholders would have no context (EA2). According to EA4, this connection provides a filter mechanism to see which challenges are relevant for a single stakeholder, which also mirrors the intention of the LSADPL. During the discussion part, EA1 stated that each stakeholder should contain a description of responsibilities. AC1 wanted to add a motivation section to the element, while EA3 and PA1 liked to include a goals section. An idea from PA1 was to aggregate the top five goals of all stakeholders and place them at a layer above the stakeholders. The user could then navigate from a goal to a stakeholder and a challenge eventually.

Evaluation of the element Initiative

The assessment of the element Initiative collected very mixed results as shown in the blue bars in Figure 4.6. The element received an average rating of 2.5 with a standard deviation of 1.3. Five respondents evaluated the initiatives to be *very valuable* because they aggregate stakeholders if one challenge is possessed by too many (EA1) and provide guidance if no clear stakeholder can be identified (DV1). EA4 assessed them as *valuable*, but could not give a clear explanation, as he found them important, but did not completely agree with the different values an initiative can have (e.g., Team Initiative, Program Initiative, ...). Another five people identified them to be of *neutral* value, because they did not understand the concept (EA2, EA3) or because they saw no value added (AC1). AC1 also said the element is too abstract and has no practice relevance, however, might be useful if new clear stakeholder can be identified. At this point, it is important to note that some interviewees gave a justification for their answer, but this justification indicates that they have

not understood the concept of initiatives. The two respondents assessing initiatives as less valuable stated that the element does not provide any added value because the content can be displayed by stakeholders (PM1, SA2). SA1 even argued that the element 'makes the whole model bad'. PA1 saw no value at all in the element, because he did not understand the concept and found the name very misleading and confusing. Hence, he would like to see the element not included in the next version. In general, many agreed to rename the element, because they misunderstood initiatives as project ideas in the first place (AC2, DV2, EA3, PA1).

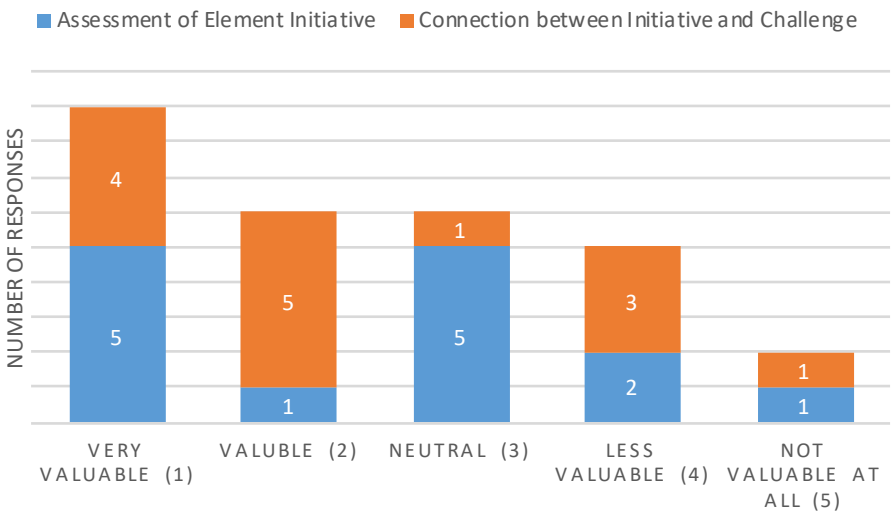


Figure 4.6.: Evaluation Results for the Element Initiative

The orange bars in Figure 4.6 display the evaluation of the connection between the elements Initiative and Challenge. The rating of the connection is slightly better as the element itself, scoring an average of 2.43 with a standard deviation of 1.29. While most participants gave an equal rating, AC1 evaluated the connection between initiatives and challenges as *less valuable*, because he did not perceive it as practical. EA3, SA1, and AC3 increased their rating from 3 to 2 but gave no justification for it. The same applies for EA4, who rated the individual element with *valuable*, but the connection with *very valuable*. The reason for this can be the increased understanding the participants achieve during the interview. Some asked which rating they gave for the element and gave the same for the connection. DV1 argued this connection to be *very valuable*, because 'challenges affect also non-identifiable stakeholders'. EA3 rated the connection as *very valuable* if the element Challenge would not have an attribute 'Scope Level' (i.e., Scaling Level).

Evaluation of the element Challenge

The element Challenge a received an average rating of 1.14 with a standard deviation of 1.3, making it the second most valuable element of the LSADPL. Despite of two respondents, everyone rated the element as *very valuable* because it is the central element of the pattern language (AC1) and provides guidance for decision making (EA1). Other points for this assessment were the fast way to find a solution based on a stakeholder (PM1), high practical relevance (AC1) and focus on what problem needs to be solved (EA4, PA1, PO1). Many respondents identified the challenges as the main part of the language because only challenges motivate patterns (DV2, SA1, SA2). However, AC2 and AC3 noticed some missing categories, for example Coaching & Training and Human Resources. While AC2 was afraid that there are too many problems and the user could not detect his specific problem, AC3 mentioned that many problems might vanish within a cluster of related problems. For these reasons, AC2 and AC3 both rated the element as *valuable*. However, AC1 emphasized that only the challenges connect people with solutions and therefore are the most important element of the pattern language. As EA1 correctly mentioned, all challenges that are irrelevant to a stakeholder can be ignored. Hence, each stakeholder can focus on his set of challenges. During the discussion section, AC1 argued that challenges should be expressed by non-agile terms and reference to a large-scale agile solution. However, the scope of the LSADPL is to capture problems that arise during the application of scaled agile methods.

Evaluation of the element Principle

Although the average rating of the element principle is 1.79, the standard deviation of 1.2 shows that the opinions on this element were divided. Figure 4.7 shows the results of three questions: the blue bar shows the ratings for the distinct element, the orange bar displays the assessment of the connection between the elements Challenge and Principle, and finally, the grey bar represents the values for the relation between Principles and Viewpoint Patterns.

The element received overall good ratings; however, PM1 argued he would not need this content at all and does not see any use case. SA1 did not understand the need to differentiate between patterns and principles, as this provides no added value and blurs the focus of the pattern language. In contrast to these opinions, EA1 explained his assessment of *very valuable* with the demand for clear guidance and EA3 agreed that principles build the foundations of teamwork. Principles provide stabilization (PA1) and are a mean for structured work (SA2). PA1 additionally explained that principles are good for new team members. PO1 argued that principles are a mean to steer someone's mindset without specifying a precise method. Because 'developers often cannot understand the demand for principles' it is necessary to write them down (AC2). The connection between challenges and principles has been assessed very similar by each participant. As PM1 did not

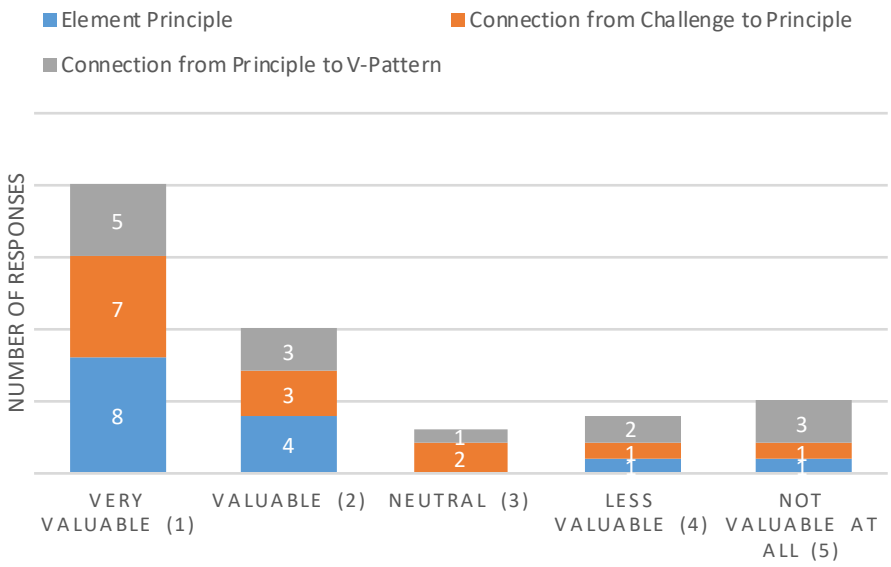


Figure 4.7.: Evaluation Results for the Element Principle

like the concept of principles, he evaluated the connection as *not valuable at all* and wanted the principles to be disconnected from everything else. The same applies for SA1, who argued that no one needs principles. The two neutral opinions on the connection came from EA4, explaining that principles are important but are generally valid and do not need to be attached to distinct challenges, and AC3, who could not give a clear answer because he required more examples. Finally, the connection between principles and Viewpoint Patterns received the most heterogeneous evaluation. EA1 and DV1 justified their assessment of *very valuable* with the ability to quantify and validate a principle as well as to provide visual feedback to the team. AC2 said the connection is 'nice to have', but he is not sure if 'it covers all cases'. All negative opinions on this connection were given by participants that also evaluated the element Principle negative before.

Evaluation of the element Coordination Pattern

The Coordination Patterns received an average rating of 1.71 with a standard deviation of 0.88. Positive opinions include the definition of a common terminology and understanding (EA4) as well as the variants section (SA2). Most participants agreed that the coordination topic is vast and needs to be separated from methods and frameworks (AC1, EA1, PA1, PM1, SA1), which will be classified as Methodology Patterns. EA2 pointed out that it is going to be hard to describe patterns in a very general manner, which is also captured by AC2, who said that coordination mechanisms are very team dependent and cannot be generalized. AC3 argued that the value of Coordination Patterns depends on the content.

The connection between Challenge and Coordination Pattern received an average rating of 1.43 with a standard deviation of 0.82. Almost all participants agreed that the connection between challenges and patterns are the core of the pattern language (AC1, AC3, EA2, SA1). The relation between Coordination Pattern and Viewpoint Pattern received very similar values with an average of 1.5 and a standard deviation of 0.82. EA2 explained that this connection might not be required, but can be helpful sometimes. DV2 analyzed that a visualization can bring more structure to a coordination meeting, therefore assessing the connection with *very valuable*. Associating Coordination Patterns with Methodology Patterns has been rated slightly worse than the connection with Viewpoint Patterns, receiving an average of 1.93 with a standard deviation of 0.97. The reasons for this include a possibly small number of connections (EA3), and missing value add (AC3, EA4). However, 71,43% of the participants rated this connection with *very valuable* or *valuable*, because methods can also be applied during coordination meetings (DV1) and it grants the pattern language more structure (PO1).

Evaluation of the element Methodology Pattern

Methodology Patterns were evaluated very positively, scoring an average of 1.5 with a standard deviation of 0.73. SA1 mentioned that methods are the core of this pattern language, as it shows solutions to recurring problems very fast (EA1, PM1). Without Methodology Patterns, the whole concept would lose its meaning (AC1). However, DV1 and EA2 explain that Methodology Patterns always require a large number of changes when applied in practice, therefore only rating it as *valuable*. This fits EA2's earlier statement that it is hard to generalize patterns so that they are easily adaptable in practice. In EA4's opinion, all Methodology Patterns can also be written as Coordination Patterns. This raises the question if the LSADPL should distinguish between Coordination Patterns and Methodology Patterns. The results of this question are displayed in Figure 4.8. Most participants agreed with our idea to split the concepts into two kinds of pattern. EA1 explained that Coordination Patterns aim at organizing people, whereas Methodology Patterns focus on what organized people do. In accordance, EA2 argued that the separation simplifies the usability of the pattern language and emphasizes its validity, which has also been mentioned by AC2. Communication and coordination are two different things and need to be separated (PA1). DV1 correctly identified Coordination Patterns as meetings and Methodology Patterns as process steps and agreed to our concept. However, 29% wanted us to merge the two concepts because every Coordination Pattern can also be written as a Methodology Pattern (EA4), they are too similar (SA1), and differentiating does not provide any value add (EA3, SA1). SA2 described the concept as 'a nice structural mechanism, but nothing more'.

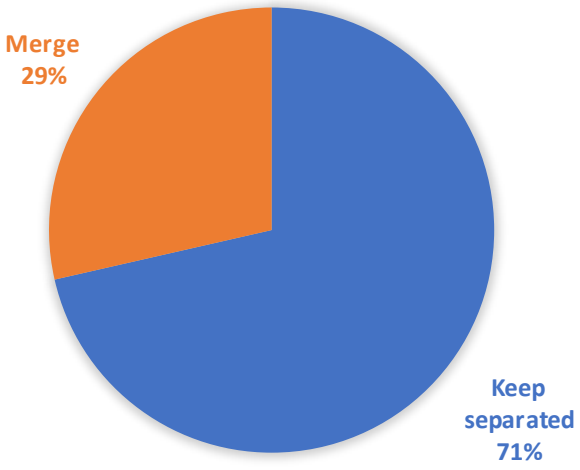


Figure 4.8.: Evaluation Results for the Distinction between Coordination Patterns and Methodology Patterns

Evaluation of the element Viewpoint Pattern

The concept of Viewpoint Patterns earned an average rating of 1.43 with a standard deviation of 0.82. 13 out of 14 participants perceived this element as *very valuable* or *valuable*, because visualizations are important (AC1) and helpful (AC2). One 'can never have enough visualizations' (DV1); hence the demand for a comprehensive, structured toolset is given (PM1). They are an important tool to convince stakeholders and team members (EA1) and to strengthen the common basis between them (PO1). SA2 has been the only one rating Viewpoint Patterns as *less valuable* because in his opinion this level of complexity cannot be found in practice and the degree of detail is too high. The link of Viewpoint Patterns directly to challenges raised a minor discussion, which is reflected in a standard deviation of 1.17, whereas the average score is 1.64. 10 participants rated the connection as *very valuable*, while only one rated it as *valuable*, two were *neutral* and one wanted the two elements to be disconnected. In SA2's opinion, a visualization is no method, if it can resolve a challenge directly. Consequently, only methodologies should be directly connected with challenges. EA3, who was neutral for this question, explained the connection itself makes sense, however 'Information Patterns are missing'. During the discussion, he also argued that KPIs do not belong to Viewpoint Patterns.

Evaluation of the element Anti-Pattern

Finally, the Anti-Patterns received an average rating of 2.0 with a standard deviation of 0.85. The distribution of the responses can be seen in Figure 4.9. It shows that no one wanted to remove the element from the pattern language. Anti-Patterns point out common traps to not fall into (EA1) and can have more information than regular patterns (AC1). Documented anti-patterns save time, because people do not need to learn their own lectures, but can learn from others failures (DV2). In contrast, AC1 mentioned that every anti-pattern can also be written as a principle, therefore limiting the value given and possibly adding more redundancy. EA2 also highlighted that anti-patterns are difficult to write.

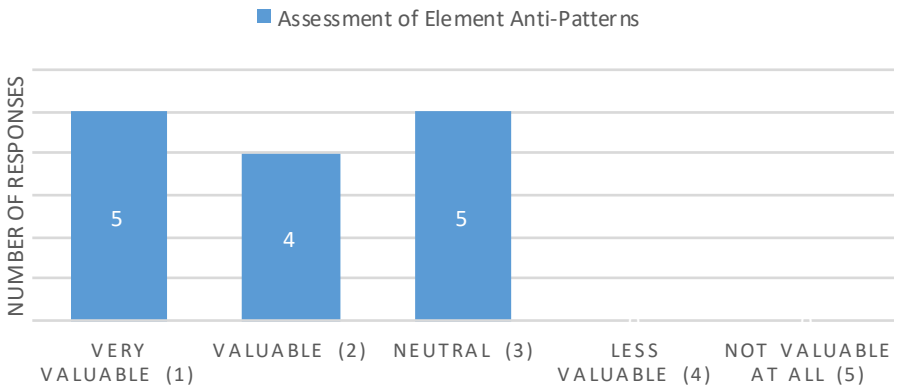


Figure 4.9.: Evaluation Results of the Element Anti-Pattern

Most neutral ratings (DV1, SA1) lead back to missing links to other elements, which is also reflected in Figure 4.10. We asked if the element should be connected to any other elements of the LSADPL. 93% of the participants answered *yes*, while only one participant wanted them separated from everything else. SA2 argued that the pattern language would be 'over-engineered' if anti-patterns would be linked to anything else. The focus needs to lie on the patterns only. However, he also mentioned that once all patterns are identified, it is possible to link the anti-patterns to stakeholders. Still, this should not be the main focus at the beginning (SA2).

Four elements were frequently named as possible elements to link the anti-patterns to:

- Stakeholder (eleven times)
- Initiative (four times)
- Challenge (nine times)
- Other Patterns (four times)

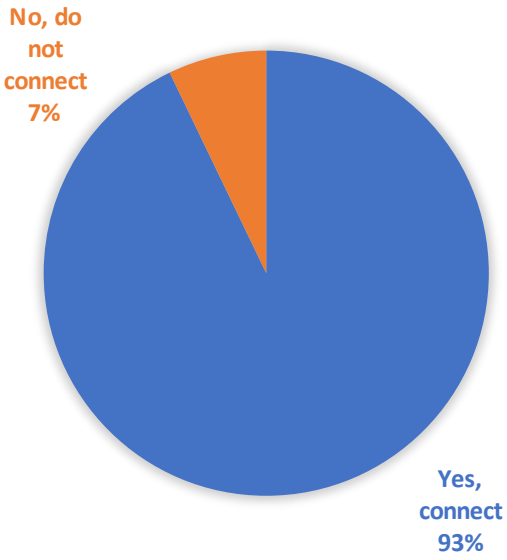


Figure 4.10.: Evaluation Results for connecting Anti-Patterns

Reasons for the connection with stakeholders included the simple match between stakeholders and their bad practices (EA3), anti-patterns always lead back to people (PM1), and each stakeholder has a set of anti-patterns (AC3). EA2 also named Stakeholder as the most valuable link, while not being sure if it is possible to map anti-patterns to stakeholders. However, most participants could not give a clear justification about why they selected a particular element to link the anti-patterns to. EA1 highlighted that Anti-Patterns should be liked to everything that makes sense, not just one single element. SA1 adds that they should be on the same level as the usual patterns.

Relevance of the Pattern Language

In the last section of the questionnaire, we asked the respondents if they would use patterns to address problems in the large-scale agile development domain (question 4.3) and how helpful those patterns are (question 4.4). Figure 4.11 shows the results for Question 4.3 and 4.4. The absence of *Less Certain* and *Certainly Not* ratings already indicates a good assessment of the usefulness of the LSADPL. Question 4.3 received an average rating of 1.79 with a standard deviation of 0.67, which shows that the pattern language has potential users. The two *neutral* ratings came from more experienced participants (EA2, AC3), arguing that they would use the pattern language only as a source of inspiration rather than to seek help. Question 4.4 has been rated slightly better with an average of 1.71 and a standard deviation of 0.59. Even though participants who would not use the pattern language said, it is good to have.

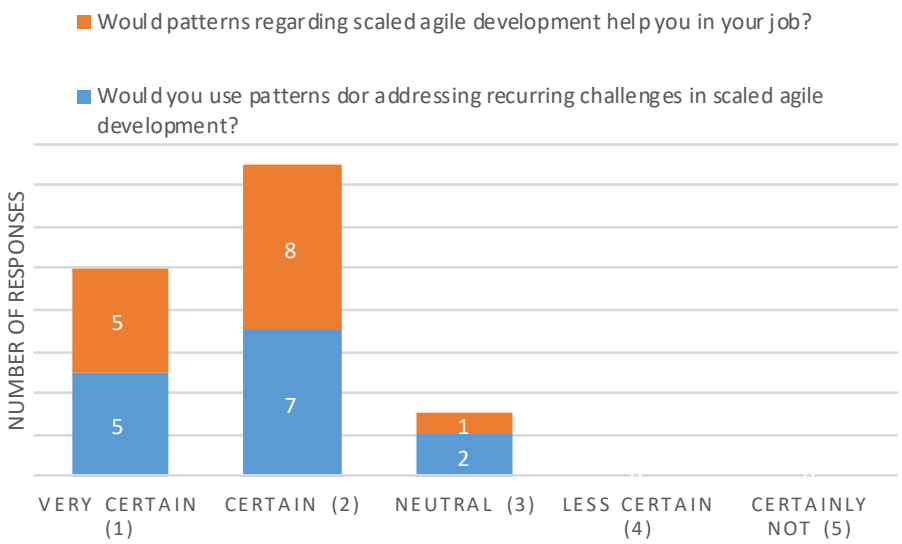


Figure 4.11.: Evaluation Results on the Relevance of the Pattern Language

Summary

The evaluation has shown that the demand for such a pattern language exists. The basic structure of the language, consisting of stakeholders, challenges, and patterns, received excellent feedback. The visualization using the interactive prototype has been appreciated by all fourteen participants in comparison to a simple document. However, a few negative points on the current version have been mentioned frequently, requiring us to prepare a second version. Further, we collected valuable feedback regarding the aforementioned discussing points. The second and also final version of the LSADPL and all changes made are explained in the next section. Figure 4.12 and Figure 4.13 summarize the average rating as well as the standard deviation of each question.

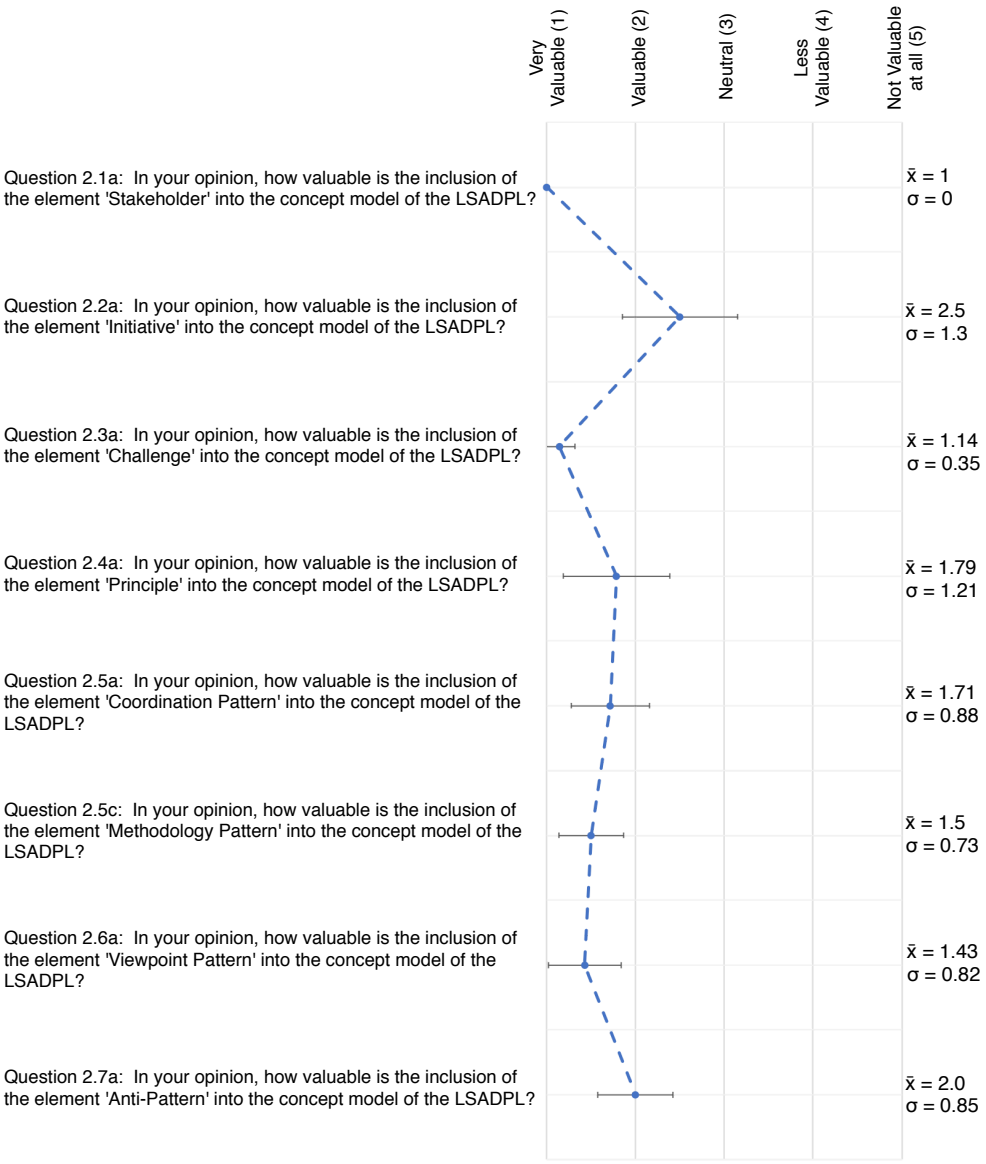


Figure 4.12.: Results for the Evaluation of each Element

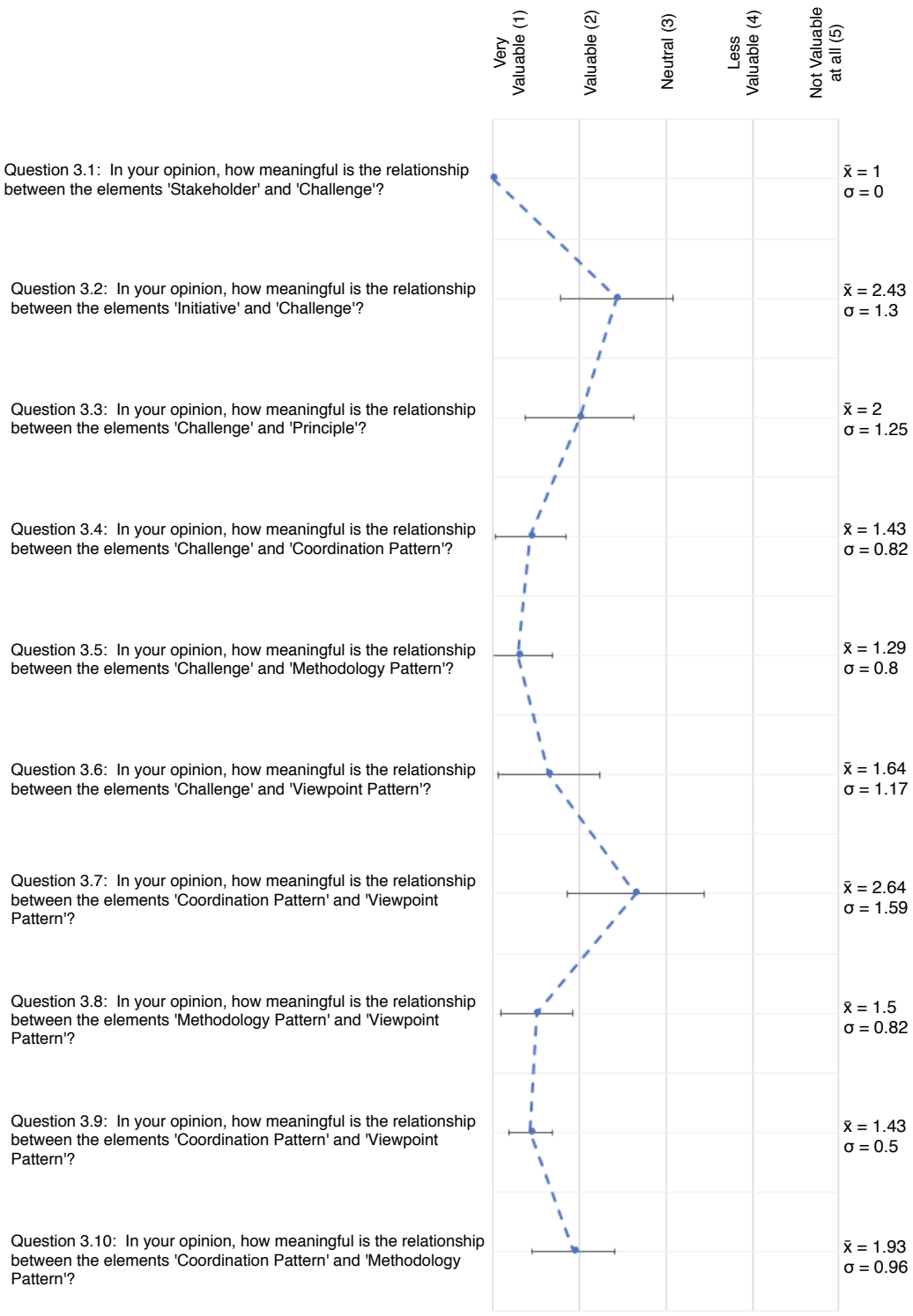


Figure 4.13.: Results for the Evaluation of each Connection

4.3. Final Version

The evaluation results lead to several changes that needed to be incorporated into the LSADPL. Due to the bad results, we decided to completely remove the element *Initiative* from the pattern language. An element that half of the users do not understand does not provide any value to the concept and therefore must not be included. The content of the element will be displayed via the 'Scaling Level' section on the element *Concern*. This element has originally been called 'Challenges', but due to the loss of the Initiatives, we decided to rename it. The term also refers to ISO 42010:2011 [45], which describes a concern as one or more stakeholder's 'needs, goals, expectations, responsibilities, requirements, design constraints, assumptions, dependencies, quality attributes, architecture decisions, risks or other issues' [45]. By using this definition, we adapt to the feedback of the evaluation to also include the goals of stakeholders. Further, we are able to address not only a stakeholder's challenge but also his tasks and responsibilities.
Another point of change is the 'Solution' section of the element *Principle*. Some participants asked us how a principle differs from a pattern, as both describe a solution to a problem. As a consequence, we removed the 'Solution' and 'Rationale' sections. The element may refer to other patterns that describe how the principle can be fulfilled or monitored. However, this was not the only change made to the element. We added another section called 'Binding Nature' to it, which marks the principle either as mandatory or as recommended. Furthermore, we removed the 'Implementation' section from the elements *Principle* and *LSAD Pattern* as there was too few difference to the 'Solution' section and the implementation is too specific for each company.
Finally, due to the majority of respondents wanting to connect the anti-patterns, we decided to connect them to the element *Concerns* as well as to *Principle* and all types of patterns. This way, the user of the LSADPL can see which anti-pattern frequently occurred for solving a specific concern. Additionally, references from an anti-pattern to a pattern allow for more revised solutions.
Summarized, the evaluation leads to the following changes in the pattern language:

- Removing the element *Initiative*
- Extending the element *Principle* with a field called 'Binding Nature'
- Removing the 'Solution' and 'Rationale' sections in the element *Principle*
- Renaming the element *Challenge* to *Concern*
- Connecting the element *Anti-Pattern* with all other elements
- Removing the section 'Implementation' from the elements *Principle* and *LSAD Pattern*

The final version is presented in Figure 4.14.

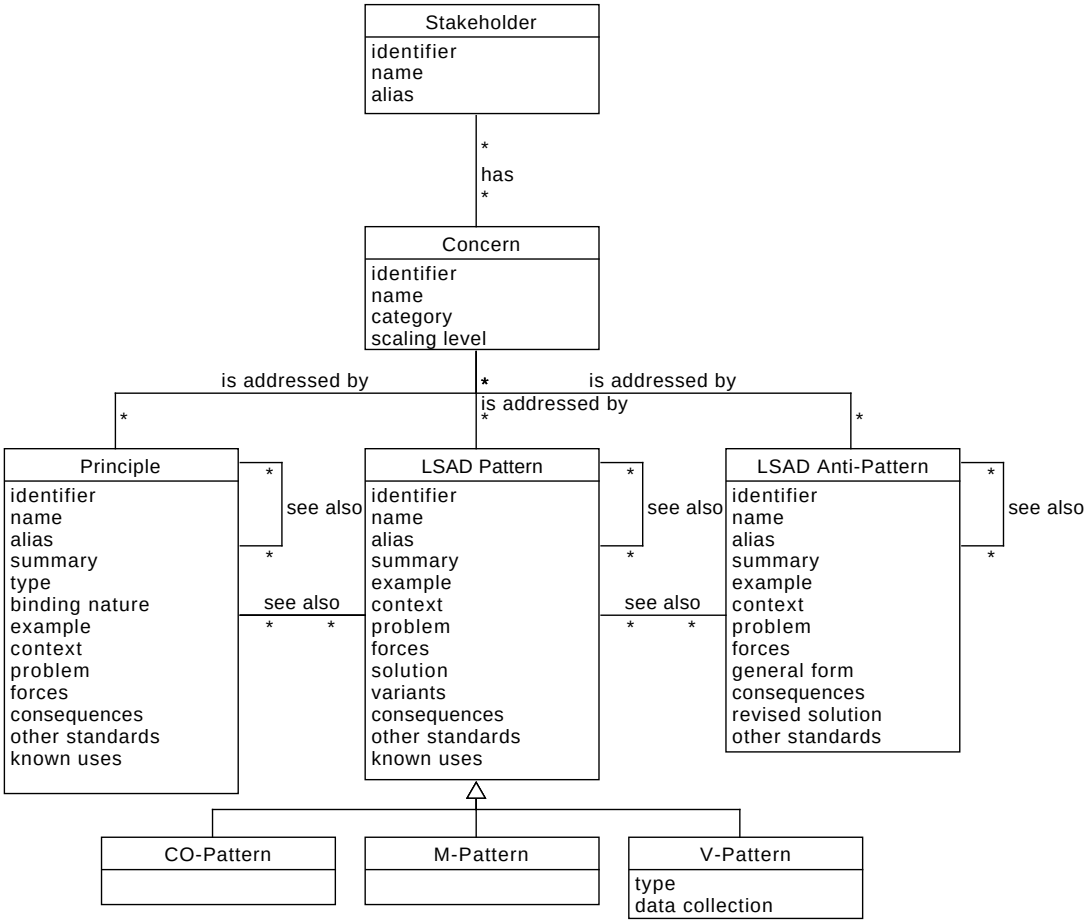


Figure 4.14.: Final Version of the Pattern Language

4.3.1. Final Elements

The elements are eventually defined as followed:
Stakeholder: remained unchanged

Concern: A concern is either a task, a responsibility, a goal, a challenge or a problem of a stakeholder. Each concern is categorized into one of the following classes: Culture & Mindset, Communication & Coordination, Enterprise Architecture, Geographical Distribution, Knowledge Management, Methodology, Project Management, Quality Assurance, Requirements Engineering, Software Architecture, or Tooling [78]. Furthermore, a concern includes the attribute *Scaling Level*, which describes the organizational level on which the concern appears. Possible scaling levels are Team Level, Program Level, Portfolio Level,

IT Organization Level, Enterprise Level. Concerns reference principles and patterns that help to solve the concern.

Principle: remained unchanged

Coordination-Pattern: remained unchanged

Methodology-Pattern: remained unchanged

Viewpoint-Pattern: remained unchanged

Anti-Pattern: An Anti-Pattern describes a common practice that has negative consequences while providing a revised solution. It can be referenced directly to a stakeholder or a concern and can refer to other patterns, which can be used in place on the anti-pattern.

4.3.2. Final Sections

Table 4.3 summarizes the final documentation templates used for the elements of the LSADPL.

Section	Description
Identifier	A unique ID for easier identification and cross-referencing
Name	A unique, concise name for the solution
Alias	How this pattern is also known as
Summary	Three to five sentences summarizing the solution
Example	A fictional story, in which the solution has been applied, can also include a description of the context and forces
Context	Circumstances under which the solution can be applied to the problem
Problem	References to one or more *Concern* entities
Forces	Describes why the problem is hard to solve
Solution	The solution to the problem within the context
Variants	Lists possible variants of the same solutions
Known Uses	List of organizations in which this pattern has been applied
Consequences	Positive consequences (benefits) and negative consequences (liabilities) when applying the solution
See Also	References to other LSAD Patterns or Principles
Other Standards	References to standards which use this pattern
Type	Categorizes the Viewpoint Pattern, e.g. Board, Blog, KPI *(only applicable to Viewpoint Pattern)*
Binding Nature	Describes if a principle is 'mandatory' or recommended' to apply *(only applicable to Principle)*
Rationale	Explains why a principle is necessary and what its intention is *(only applicable to Principle)*
General Form	Problematic solution to the problem *(only applicable to Anti-Pattern)*
Revised Solution	How the problem should be solved instead *(only applicable to Anti-Pattern)*

Table 4.3.: Final Documentation Template of the Large-Scale Agile Development Pattern Language

5. Identification of Recurring Concerns and Best Practices

The previous chapter described the entire development process of the LSADPL. The next step is to identify recurring concerns and best practices for Agile Coaches and Scrum Masters and document them as patterns according to the previous template. This chapter presents all results from the second interview series. It will first provide an overview of the methodology (Section 5.1) and list all identified concerns in Section 5.2.1. Afterward, duplicates are identified and findings on the concerns are summarized (Section 5.2.1). In the second part, this chapter presents all identified patterns and pattern candidates in Section 5.3.

5.1. Methodology

For identifying recurring concerns and best practices, we used semi-structured interviews. The outline for an interview is shown in Figure 5.1. The goal of the interview series was to analyze the practical relevance of existing concerns found in the literature on the one side, and on the other side to identify new concerns, which have not been mentioned in literature. Additionally, patterns for large-scale agile development should be identified.

An interview was scheduled for 90 minutes in total and divided into the following phases:

1. **Introduction**: The interviewers introduced themselves to the respondent and presented the topic as well as the intention of the interview. The prototype of the pattern language has been shown and explained shortly.
2. **Identify I**: The respondents were asked to name their top three concerns as an Agile Coach or Scrum Master. They should shortly describe them as well as categorize them and assign them to one or several scaling levels.
3. **Describe I**: For each concern named in *Identify I*, the respondent was asked to explain his solution to the concern.
4. **Identify II**: The respondents were given a list with all existing concerns and had to tick off the ones they also face.
5. **Describe II**: The respondents could choose any concerns of the list provided in *Identify II* and explain their solution to the concern.

Two researchers participated in each interview, both taking notes and asking questions when required. One researcher was responsible for timekeeping as well as recording the interview.

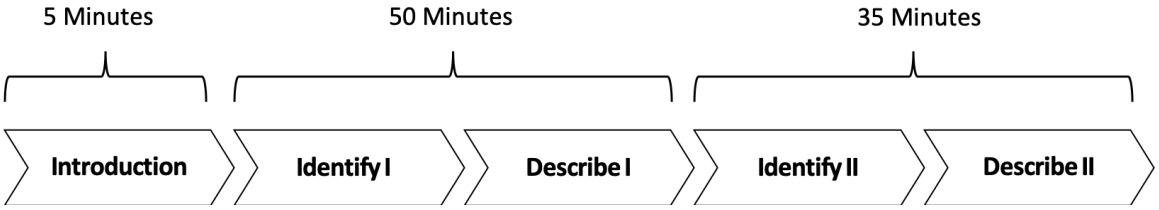


Figure 5.1.: Outline of the semi-structured Interviews for identifying Patterns

After all interviews have been conducted, the documented patterns, anti-patterns and principles have been sent to the respecting participants for another check. Along with the patterns, we also sent them another questionnaire containing all identified concerns. The participants were asked to mark all the concerns they also face. Table 5.1 summarizes all participants, their experience with large-scale agile development as well as their companies' experience with large-scale agile development, their size, and sector.

In contrast to the first interview series on the evaluation of the pattern languages, these interviews required very skilled participants. All respondents have experience with large-scale agile practices of at least one year, which increases the validity of the results. Recurring concerns and practices can only be observed if someone has been working in a domain for a while. Some participants have also participated in the first interview series, described in section 4.2. However, the alias has not been reused for this section. Hence, AC1 in this chapter is not the same person as AC1 in section 4.2.

5.2. Findings on Recurring Concerns

5.2.1. Concerns identified by Agile Coaches and Scrum Masters

The first part of the interview was focused on the personal concerns each participant has. Each of them should name and explain their top three concerns. However, some could not think of three, and others mentioned more. The following list describes all concerns mentioned by the participants and describes them according to the interviewees' explanations.

- **C-80 Improving Personnel, Product & Process Excellence**: The organization needs to define a set of roles and ground them in the corresponding flat hierarchies ('Personnel Excellence'). Processes need to be restructured and simplified in order to enable an agile work style ('Process Excellence'). Instead of thinking in projects, the enterprise needs to think in products, requiring appropriate slicing of a product to

No.	Alias	Role	Own Experience	Company's Experience	Sector	No. of Employees
1	AC1	Agile Coach	3 - 6 years	More than 6 years	IT, Technology	5.001 - 10.000
2	AC2	Agile Coach	3 - 6 years	1 - 3 years	Production	100.001 - 200.000
3	AC3	Agile Coach	1 -3 years	3 - 6 years	Service Industry	51 - 250
4	AC4	Agile Coach	More than 6 years	1 - 3 years	IT, Technology	51 - 250
5	AC5	Agile Coach	3 - 6 years	Less than 1 year	Service Industry	501 - 1.000
6	AC6	Agile Coach and Scrum Master	More than 6 years	3 - 6 years	IT, Technology	100.001 - 200.000
7	AC7	Agile Coach	3 - 6 years	3 - 6 years	Service Industry	51 - 250
8	AC8	Agile Coach	1 - 3 years	1 - 3 years	Finance, Insurance, Real Estate	> 200.000
9	AC9	Agile Coach	1 - 3 years	1 - 3 years	Retail	50.001 - 100.000
10	AC10	Agile Coach	More than 6 years	1 - 3 years	Production	> 200.000
11	AC11	Agile Coach	3 - 6 years	1 - 3 years	Service Industry	11 - 50
12	AC12	Agile Coach	3 - 6 years	3 - 6 years	Retail	50.001 - 100.000
13	AC13	Agile Coach	3 - 6 years	3 - 6 years	Service Industry	11 - 50

Table 5.1.: Interview Respondents of the second Interview Series

enable the effective organization of teams to slices ('Product Excellence'). It is the Agile Coaches responsibility to create a road map for establishing the three types of excellence and guide the organization through this agile transformation. AC1 classified this concern as a 'Methodology' concern on Enterprise level. This concern is a very high-level concern as it mirrors the whole agile transformation of an organization.

- **C-81 Enable Change from Process- to Product-Orientation**: The value stream within an organization needs to be altered from a process-oriented to a product-oriented view. The organization needs to get a rather customer-centered value stream. It is the task of the Agile Coach to trigger and guide this change. AC1 categorized this concern as a 'Methodology' topic on IT Organization and Enterprise level.

- **C-82 Introducing Agile Principles in a Planning-Oriented Organization**: The organization is used to have detailed plans for everything they do, including long-term projects and goals. If change occurs, the plan needs to be altered accordingly. The same applied to the agile transformation: people expect everything to be planned in advance. However, the transformation consists of a lot of 'inspect and learn' steps, making it impossible to set up a detailed top-down road map from the beginning on. People need to accept this fact by internalizing an agile mindset, adhering to agile principles like self-organization and continuous improvement. They need to understand that they are part of the transformation process. AC2 categorized this concern as 'Culture & Mindset' on IT Organization level.

- **C-83 Dealing with External Developers**: Because of cost reduction and missing know-how, development is often outsourced to external providers. However, this brings up several challenges especially if the project management is still internal. Along with many communication issues, the internal and external team members need to clear model to work together, especially if they are distributed. Moreover, legal requirements increase the complexity when it comes to assigning work and paying the external developers. These issues address the category of 'Communication & Coordination' on IT organization level.

- **C-84 Initiating People to work with Agile Principles**: Agile practices not only consist of a set of methods but heavily rely on principles and values (see Section 2.1. An organization needs to internalize this, which is very hard to process-oriented people. In traditional enterprises, where everything is defined by a process, people struggle with not having a clear recipe on how to apply agile practices. They expect a clear framework and a rollout plan. The Agile Coach needs to clarify that they will not get any of these and train them in working in agile principles. This concern has been assigned to 'Culture & Mindset' on team level.

- **C-85 Self-Awareness as an Agile Coach**: During the agile transformation, many emotions are triggered in people's minds. Therefore, it is important that the Agile Coach has a high degree of authenticity and know himself well. He needs to insist on his opinions and viewpoints. Therefore, he has to understand what kind of a person he is and how he is noticed by others. The Agile Coach should not mimic something he has seen somewhere else but be himself instead. This concern has been categorized to 'Culture & Mindset' It does not depend on who the Agile Coach coaches, therefore this concern arises on all levels.

- **C-86 Dealing with Emotional Consequences of Agile Transformations**: People need to understand that change brings emotional consequences that they cannot suppress because of natural instincts. The Agile Coach has to make it clear that emotions are fine and a natural thing that affects everyone. It is important that the Agile Coach does not miss this human factor during the agile transformation. This concern is mapped to 'Culture & Mindset' and 'Communication & Coordination' on all scaling levels.
- **C-87 Patience during the Agile Transformation**: Agile transformations take time and require much patience. The Agile Coach needs to constantly remind the organization that it takes time to change not only process but also mindsets. He needs to make clear that change is a continuum and things do not change overnight. Again, this concern has been categorized as 'Culture & Mindset' on all scaling levels.
- **C-88 Building an Agile Organization around Norms and Standards**: Each team uses different technologies and practices. An agile organization needs to cope with this heterogeneity and build up a shared knowledge base. Standards and norms not only need to be defined but also elaborated and implemented. This concern has been categorized as a 'Tooling' concern on program level.
- **C-89 Establishing Equality among Cross-Functional Teams**: In cross-functional teams, each team member typically has a wide skill set and is able to fulfill many different tasks. Consequently, if a team member calls in sick, another team member takes over his work, because he already has the required skills and knowledge. Still, each team member has his particular field of expertise, in which he is outstanding. This turns into a negative concern if this team member always gets the tasks focusing on his specialization and not the team member with free capacity. It gets even worse if this applies not only to one team member but to the whole team, dismissing the concept of a cross-functional team. This concern is classified as a 'Project Management' concern on team level.
- **C-90 objective Measurement Methods**: According to AC5, every questionnaire is biased to some degree, consequently being very subjective. Depending on the parameters included, KPIs always lead to a certain direction the creator wants to achieve. There is a demand for monitoring the status of a team from an objective view. This concern has been mapped to the category 'Tooling' on team level.
- **C-91 Demonstrate Value Add of Agile Methods**: This concern tightly relates to C-46 '*Dealing with closed mindedness*' as people, who oppose agile methods, first need to understand their value add. They need to internalize why agile methods are important for the success of the organization instead of only understanding how these methods are applied. Therefore, it is the Agile Coach's responsibility to explain and practically demonstrate the value add of agile methods. Categorized to 'Culture & Mindset', this concern occurs on all scaling levels.

- **C-92 Coordination of Multi-Vendor Teams**: In contrast to C-83 '*Dealing with external developers*', which is more generic and only deals with the fact that there are external developers within an agile team, this concern specializes on multiple external teams from different vendors. This brings more complexity to the product, more communication and coordination issues, and special legal challenges. The concern has been classified as 'Communication & Coordination' on Program level.
- **C-93 End-to-End usable functionality in one iteration implemented**: Traditional project management life cycles start with an analysis of the requirements, followed by a design phase, the implementation and conclude with testing and delivery. Agile teams often work with user stories, meaning they need to undergo all of these phases within one iteration. This gets even more complicated if different systems are affected. Figure 5.2 shows the concept based on traditional approaches and the optimal agile approach. Teams with a strong traditional background often use the upper slicing methodology. The Agile Coach needs to train the team to apply the lower slicing approach.

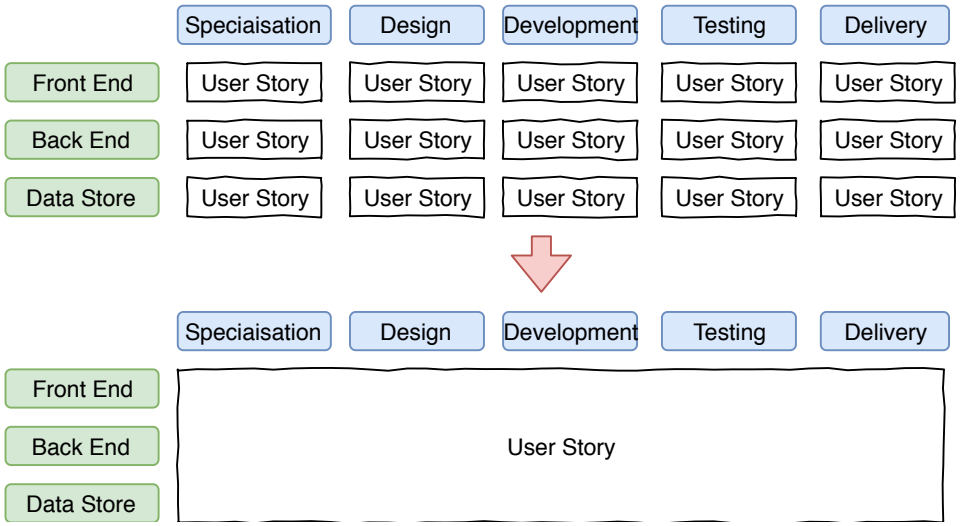


Figure 5.2.: Comparison of Slicing User Stories in Traditional and Agile Approaches

- **C-94 Understanding the Demand for Becoming Agile**: Both, stakeholder and team members need to understand why the agile transformation is necessary and how it is done in general. Moreover, the Agile Coach has to explain when agile practices can be applied and under which circumstances agility does not make sense. AC7 categorized this concern as 'Culture & Mindset' on all scaling levels.
- **C-95 Lacking Orientation because of Missing Leadership**: Self-organization also requires a minimum degree of leadership. People are afraid during the agile transformation, because of much change. AC7 mentioned the example that a Software

Architect loses his role during the introduction of Scrum and becomes only a part of the development team. A part of the leadership team should provide orientation to these people and mediate between 'stability and instability' (AC7). However, these mediators often are external agile coaches and not internal managers. Therefore, the agile coach has to prepare the organization for the time after his retirement to prevent missing orientation. This concern is also categorized as 'Culture & Mindset' on all scaling levels.

- **C-96 Decisions on higher Levels reach lower levels**: Decisions that have been made on a higher hierarchy level need to trickle down to every team member, which is hard to establish. Because all members of an organization need to contribute to a common goal, everyone needs to be informed about the strategy. This concern has been mapped to 'Communication & Coordination' on all scaling levels.
- **C-97 Frameworks on all Hierarchy Levels**: An organization needs to develop a framework that captures agility on all hierarchy levels. This framework can never be a standard framework (e.g., SAFe®), mapped to the organization one on one. An Agile Coach or a team of Agile Coaches, in combination with internals, need to tailor frameworks such that they fit the organization's strategy and processes. Categorized to 'Knowledge Management', this concern occurs on all scaling levels.
- **C-98 Common Understanding of Agility**: Each member of an organization should acquire the same understanding of agility and should use the same vocabulary. AC8 has chosen 'Culture & Mindset' as category and identified this concern on all scaling levels.
- **C-99 Missing Understanding of Roles**: Everyone needs to know not only what his role's responsibilities are, but also understand other roles meaning within the organization. This becomes an even bigger issue if the performance of fulfilling a role is coupled to financial bonuses. This concern has been categorized as 'Methodology' on team level.
- **C-100 Higher Soft skill Requirements**: Due to more communication with more people, soft skills are much more important in agile organizations. Agile Coaches need to find an effective training method to increase the soft skills of each team member. In AC8's opinion, this concern does not fit into any category. Therefore, he would like to introduce a new category 'Training'. The scaling level this concern occurs on is 'Team'.
- **C-101 Dealing with increased number of Coordination Meetings**: Several recurring meetings are defined by the framework the organization uses. However, the organization had its recurring meetings even before the agile transformation. This concern addresses the problem that old coordination meetings are not abandoned and become redundant to the new agile meetings. Consequently, people have no time to

develop as they are always in meetings. This is a 'Communication & Coordination' concern on all scaling levels.

- **C-102 Motivating Leadership to talk to Teams**: The Product Owner should not be the only one who talks to the team and communicates the higher-level decisions as well as the common product vision. The leadership also needs to know what is happening within the teams and how they are performing. This concern has been categorized as 'Culture & Mindset' on Team level.
- **C-103 Establishing A Common Product Vision**: Everyone within the organization needs to know the big picture and the common goal everyone is working towards. Many distinct teams need to know what other teams are doing and if they are still working towards the right direction. It is the Agile Coach's task to bring the teams together and create a common road map. Without a common focus, it is almost impossible to get the teams to work towards value maximization. This concern is categorized as 'Project Management' on Portfolio, Program and Team level.
- **C-104 Definition of the Product Owner Role**: Especially in large companies, ownership regarding parts of the product is very distributed. In the beginning, it is hard to identify who the Product Owner is and what responsibilities he has. It is difficult to understand who is able to take which decisions. Consequently, this concern does not target the definition of tasks and responsibilities of the Product Owner role. Rather, it points out the difficulty to identify the Product Owner. This concern has been categorized as 'Project Management' on Portfolio level.
- **C-105 Behavior of Middle Management**: The Middle Management tends to a 'Command and Control' leadership style. They want to be as good as possible and communicate that they are better than other teams, making them act very egocentric and not agile. Managers have few knowledge about agility and the expectations of the team. This concern is part of the 'Culture & Mindset' category and occurs on IT Organization level.
- **C-106 Agile Program Management**: A program typically has several milestones that need to be reached at a certain point in time. This is contrary to the agile approach. AC10 explains that the point in time will definitely come, but it is not known in advance if the status of the program will be satisfactory. This aspect of agile program management needs to be accepted by the organization. However, managers fear this degree of uncertainty for a whole program. This concern has been categorized as 'Project Management' on Program level.
- **C-107 Acknowledging the Demand for Change**: Before taking changing action, the organization needs to understand and acknowledge that they need to change not only their ways of working but also their mindset. This concern has been categorized as 'Culture & Mindset' on all scaling levels.

- **C-108 Transforming an Impediment into a Learning Backlog**: An impediment is everything that hinders the team from creating value to the customer. The organization has to understand that each impediment is a chance to learn to improve and learn. Again, this concern is mapped to 'Culture & Mindset' on all scaling levels.

- **C-109 Learning to Improve as an Organization**: Not only individuals or teams must improve, but also the organization as a whole. They need to decide on things jointly and work on topics together. Again, this concern is mapped to 'Culture & Mindset' on all scaling levels.

- **C-110 Establishing an Agile Mindset**: Categorized as 'Culture & Mindset' on all levels, faces the general challenge on how to establish an agile mindset within an individual or a team.

- **C-111 Coaching on Higher Management Level**: Higher management is not part of the classic agile frameworks, like Scrum, but has to be considered in scaled agile methods. This concern not also addresses how to coach managers, but also how to achieve the mandate to coach managers. It has been categorized as 'Culture & Mindset' on Enterprise, Team and IT Organization level.

- **C-112 Identifying Dependencies between Teams on Cross-Domain Level**: Categorized as an 'Enterprise Architecture' concern on Enterprise, IT Organization, and Team level, this concern refers to technical dependencies between teams on cross-domain level. The focus only lies on identifying and managing those dependencies.

- **C-113 Creating the Agile Mindset for Higher Management**: Given the different mindsets between team members, working directly for the customer, and managers, who are known to be more resistant to change and new methods, this concern addresses the creation of an agile mindset for higher management. The two mentioned stakeholders need different approaches, therefore demanding a distinct concern. It has been categorized as 'Culture & Mindset' on Enterprise and Portfolio level.

- **C-114 Understanding that Organizational Development is as Important as Product Development**: Not only the product development needs to be transformed to work in an agile way, also the organization has to chance. Processes and mindsets need to be changed to make agile methods work. This concern addresses the category of 'Culture & Mindset' on Enterprise and Portfolio level.

- **C-115 People are overloaded with work**: Although agile practices propose only to occupy people by approximately 80%, everyone in the organization is chronically overloaded with work. This leads back to incorrect agile practices and has been consequently categorized as a 'Project Management' concern on all scaling levels.

Merging Duplicates and Removing Incongruous Concerns

Due to the structure of the interview, recipients did not know the existing concerns in advance and therefore, mentioned duplicates of existing concerns during the first phase. We merged those concerns in order to remove the number of concerns such that the LSADPL becomes more usable for the user. Further, some concerns needed to be removed, because they did not fit from a pattern-writing perspective. First, C-98 *Common Understanding of Agility* is already covered by C-59 *Establishing a common understanding of agile thinking and practices*. Both include the definition of frequently used terms and methods as well as sharing knowledge. All pattern candidates for C-98 that have been mentioned during the interview with AC 8 will be mapped to C-59. Further, C-99 *Missing Understanding of Roles* is a duplicate of C-56 *Defining clear roles and responsibilities*. Even though the wording is different, the content of both concerns is the same. Moreover, C-101 *Dealing with increased number of Coordination Meetings* can be merged with C-49 *Dealing with increased efforts by establishing inter-team communication* as both address the increasing number of meetings when applying scaled-agile frameworks. Concern C-80 *Improving Personnel, Product & Process Excellence* has a too broad scope and is therefore split into three other concerns, shown in Figure 5.3. We decided to remove C-82 *Introducing Agile Principles in a Planning-oriented Organization* because from a pattern perspective, the fact that the organization is planning-oriented is part of the context and not part of the problem. Similar, we removed C-108 *Transforming an Impediment into a Learning Backlog* is not a problem itself, but rather a solution to other problems, for example, C-39 *Establishing a Culture of Continuous Improvement*. Other duplicates include C-105 *Behavior of Middle Management*, which is mapped to C-47 *Dealing with higher-level management interference*. Further C-107 *Acknowledging the Demand for Change* has been mentioned before and is a duplicate of C-94 *Understanding the Demand for Becoming Agile*. Another management concern C-113 *Creating the Agile Mindset for Higher Management* is mapped to C-111 *Coaching on Higher Management Level*, as C-113 is included within C-111. As the intention of both concerns is the same, C-114 *Understanding that Organizational Development is as Important as Product Development*, this concern is mapped to C-109 *Learning to Improve as an Organization*. The last concern mentioned during the interviews, C-115 *People are overloaded with work* is a duplicate of C-16 *Dealing with increasing Workload of Key Stakeholders*. Finally, not only this thesis not only checks for duplicates between the new concerns and the ones found in the literature for Agile Coaches and Scrum Masters but also checks for duplicates within the concerns of all other stakeholders identified by [78]. This lead to the identification of C-103 *Establishing a Common Product Vision* being a duplicate of the concern *Sharing Common Vision* by the Program Manager and the Product Owner.

Figure 5.4 summarizes the categories of the final list of concerns, divided into concerns found in the literature [78] and concerns identified during the interviews as part of this work. It reveals that most concerns address 'Culture & Mindset'-related topics, which fits the literature review, which also identified 'Culture & Mindset' as the largest category. In contrast, the second largest category 'Geographical Distribution' is only relevant in litera-

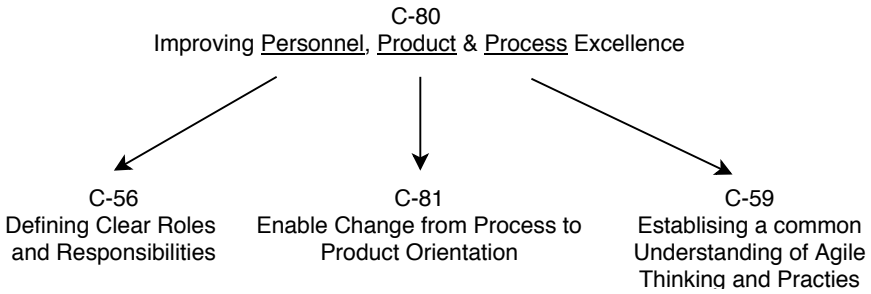


Figure 5.3.: Splitting C-80 into three minor Concerns

ture and none of the participants mentioned problems in this area.

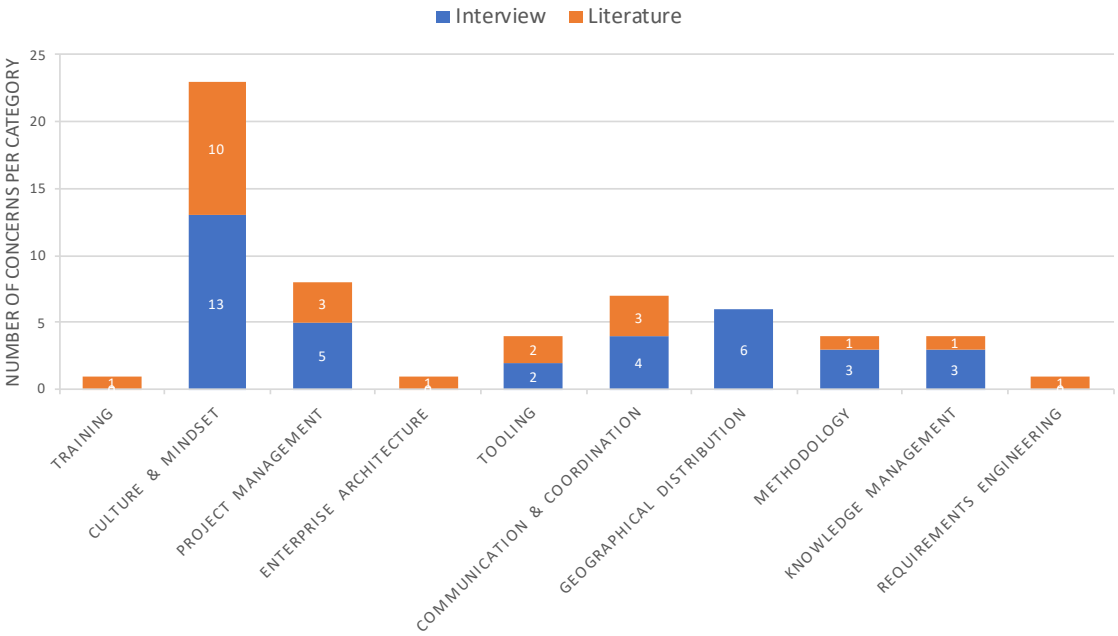


Figure 5.4.: Distribution of Categories among all identified Concerns

As in large-scale agile development, concerns do not only occur on team level, Figure 5.5 presents the distribution of scaling levels across the identified concerns. While each concern can have exactly one category, it can have multiple scaling levels, as it can also occur on different levels. The graph shows that only 26% of all concerns occur on team level. However, they can also occur on other scaling levels, like program or enterprise level. The graph also reveals that 74% of all concerns occur on higher levels, which indicates

The final documentation of concerns is provided in Appendix B.1, which is one of the

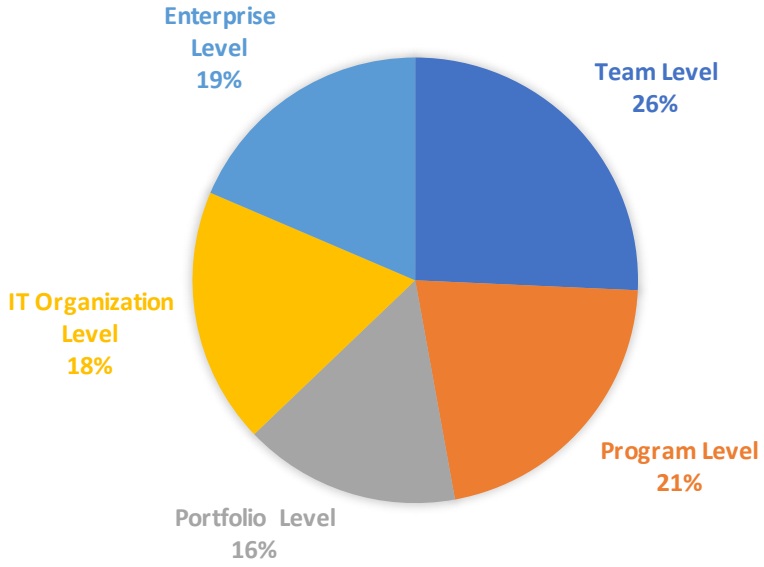


Figure 5.5.: Distribution of Scaling Levels among newly identified Concerns

primary artifacts developed by this thesis.

5.2.2. Identification of Recurring Concerns

This thesis is also aims at identifying which concerns recur throughout different companies. Therefore, we asked every interview participant to fill out a questionnaire providing a list of all identified concerns, including the ones from the literature review. The results can be found in Figure 5.6 and Figure 5.7. Both figures include all concerns for Agile Coaches, Scrum Masters, and Initiatives identified by Uludağ et al. [78] as well as all concerns identified in the previous subsection. Duplicates are already removed from the graphs. The graphs show how many respondents marked a concern as 'Yes, I have this concern'. According to the Rule of Three [17], all identified concerns can be declared as *recurring*.

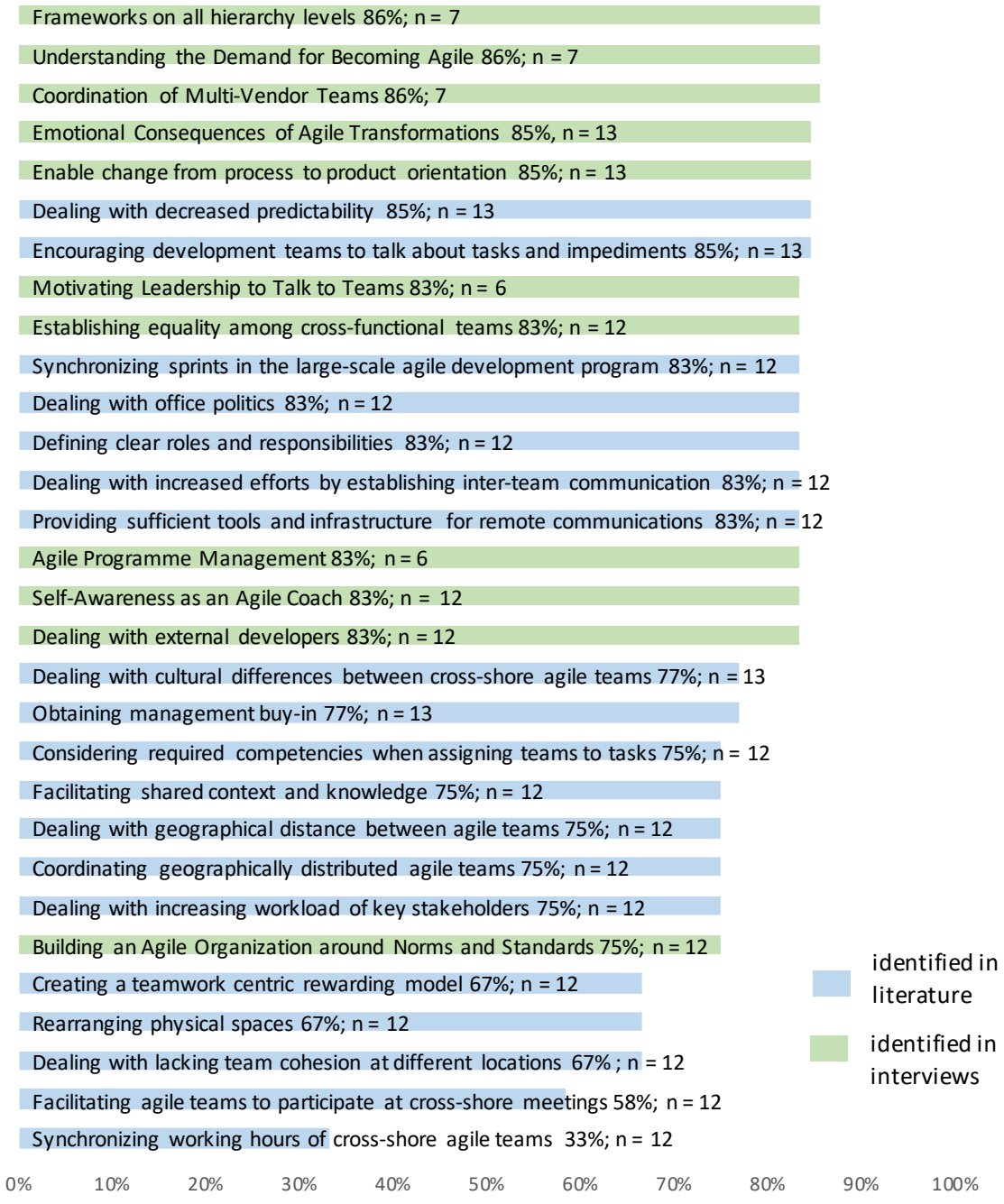


Figure 5.6.: Results of the Questionnaire on identifying recurring Concerns

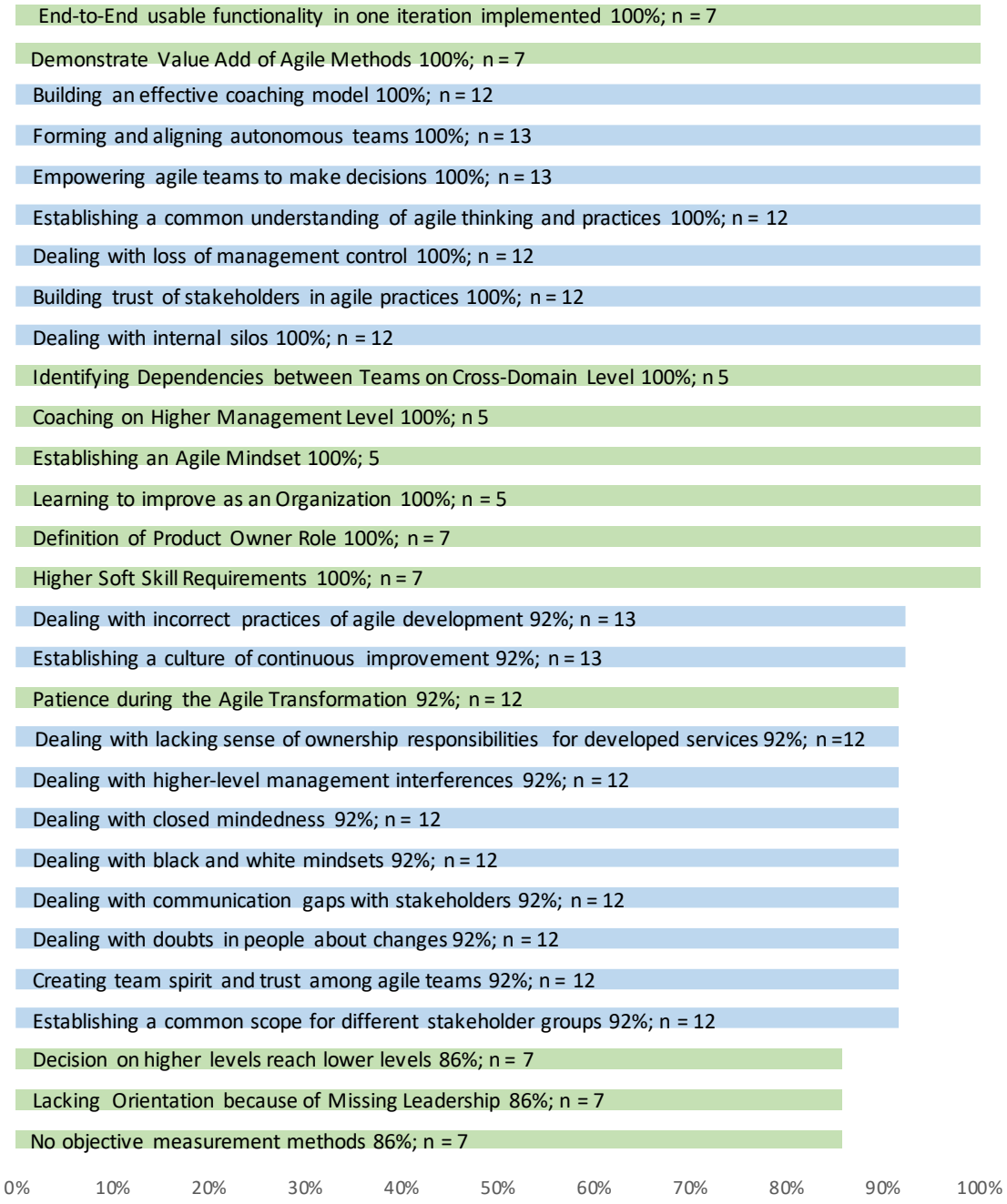


Figure 5.7.: Results of the Questionnaire on identifying recurring Concerns continued

5.3. Findings on Patterns

During the interviews we identified 76 possible pattern candidates, which divides into 21 M-Patterns, 18 V-Patterns, 14 CO-Patterns, 12 Principles, and 10 Anti-Patterns as shown in Figure 5.8.

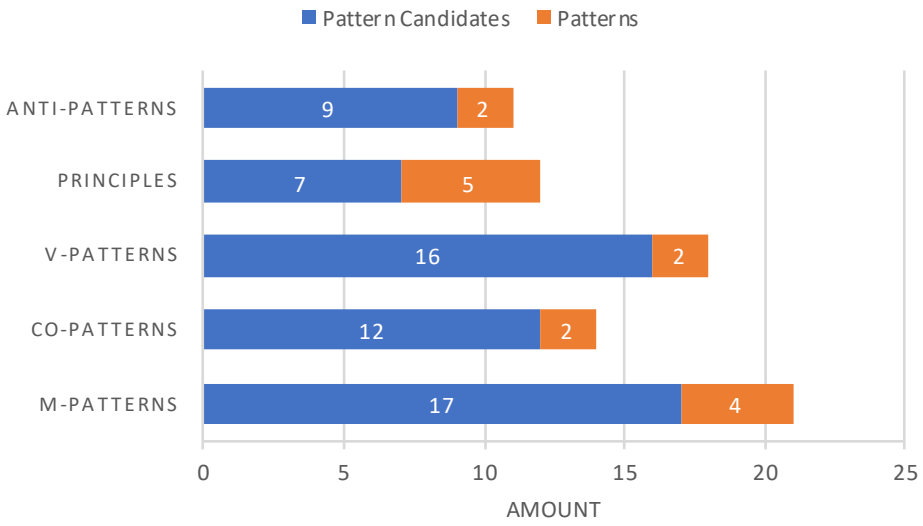


Figure 5.8.: Amount of Patterns and Pattern Candidates per Category

According to the Rule of Three, a pattern candidate becomes a proven pattern, if it occurs in three different organizations [18]. Thus, we identified the following patterns:

1. CO-01: SUPERVISION
2. CO-02: COMMUNITY OF PRACTICE
3. M-01: ROLE FOCUS
4. M-02: OBJETIVES AND KEY RESULTS
5. M-03: EMPOWERED COMMUNITY OF PRACTICE
6. M-06: GLOBAL IMPEDIMENT PROCESS
7. V-01: GLOBAL IMPEDIMENT BOARD
8. V-02: GOOD PRACTICE NEWSLETTER
9. P-01: PUBLISH GOOD PRACTICES

10. P-02: EXPLAIN MEETING PURPOSE
11. P-03: CELEBRATE EVERY SUCCESS
12. P-04: CONSENSUS-BASED DECISIONS
13. P-05: PILOTING
14. A-01: DON'T USE FRAMEWORKS AS RECIPES
15. A-06: DON'T OVERSHOOT COORDINATION MEETINGS

Despite the identified patterns, we found a large number of pattern candidates, i.e., concepts that were not mentioned at least three times. All pattern candidates are shown in Figure 5.10. These concepts are currently no confirmed pattern; however, they can become a pattern during a later point of research, when observing other stakeholders. Figure 5.9 shows the relationships between all identified patterns, principles and anti-patterns. The graph only includes concerns that relate to a solution. The following subsections document five patterns, one for each category. All remaining patterns can be found in Appendix C, while short descriptions of all pattern candidates can be found in Appendix D. For anonymity, the 'Known Uses' sections only contains aliases of the participated companies.

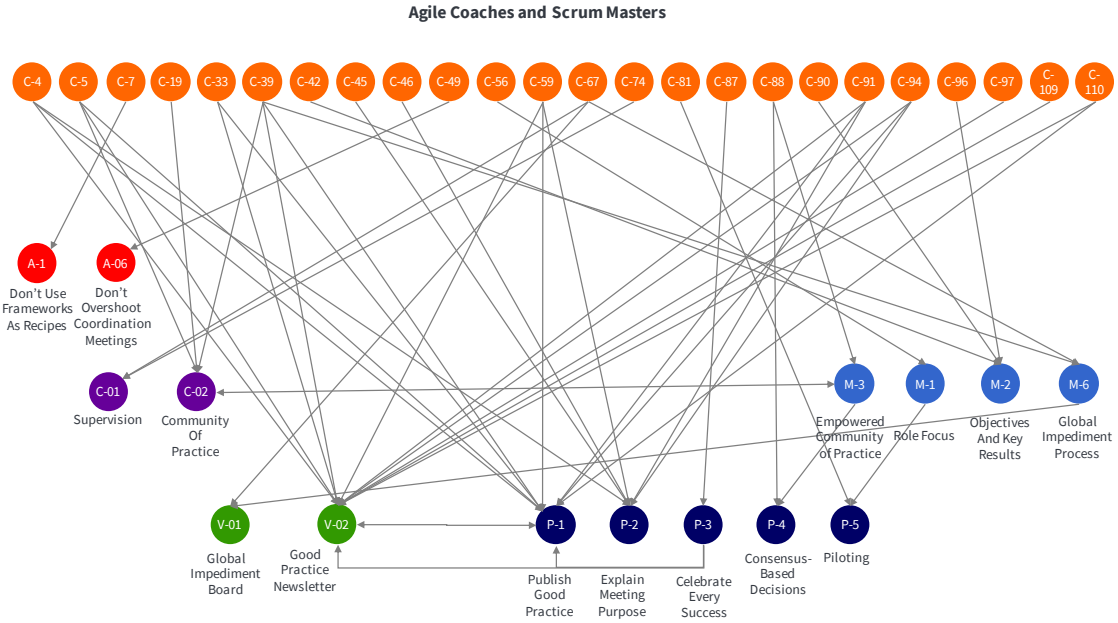


Figure 5.9.: Relationships between identified Concerns, Patterns, Principles and Anti-patterns

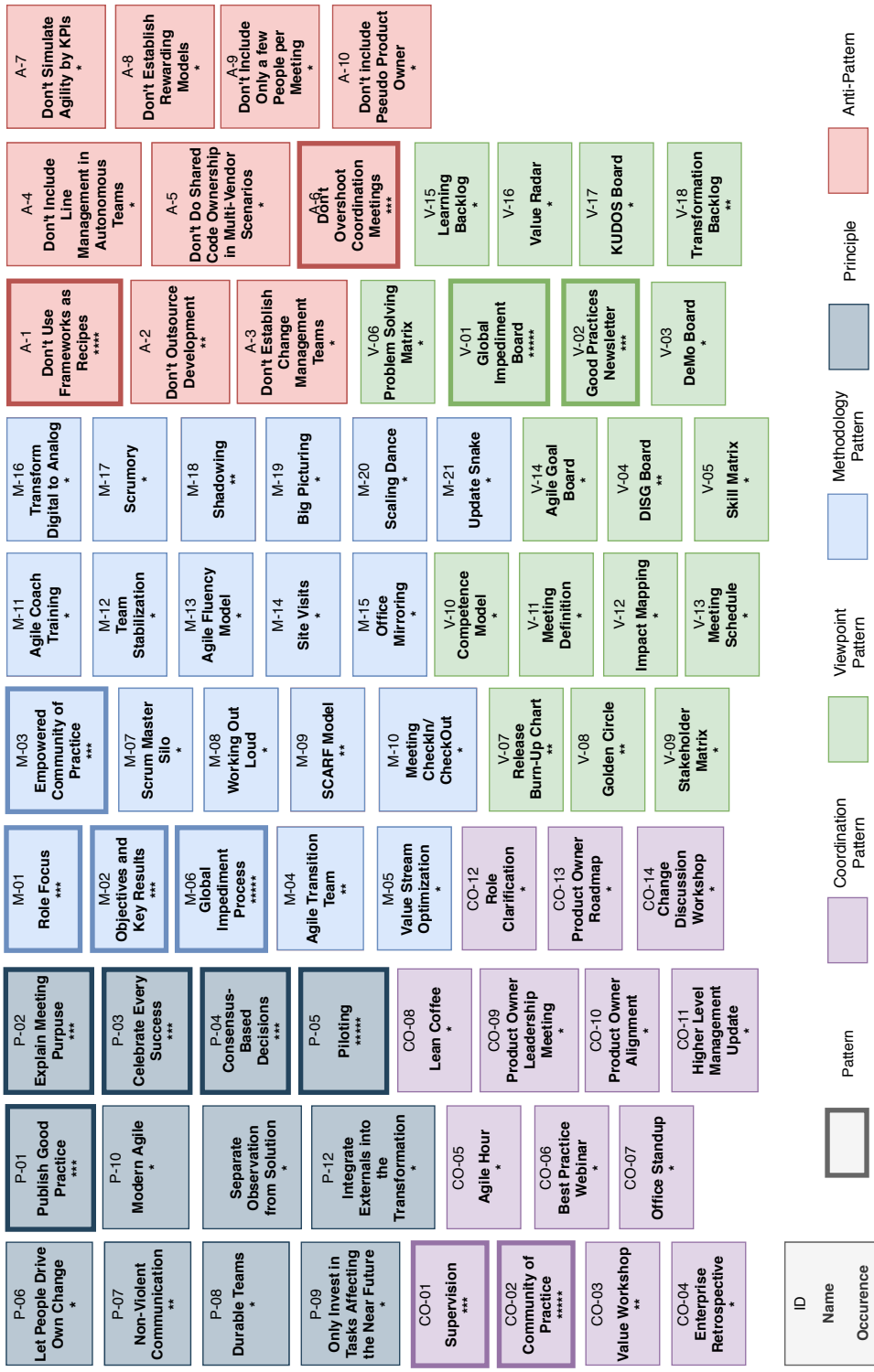


Figure 5.10.: Overview of all identified Pattern Candidates

5.3.1. Community of Practice

Pattern Overview	
ID	CO-02
Name	COMMUNITY OF PRACTICE
Alias	
Summary	To facilitate knowledge sharing, Communities of Practice can be set up. Those communities are regular meetings, in which participants can freely discuss practices and share their experience. Communities of Practice always focus on one domain, for example, Leadership, Architecture or Testing.

Example
During the transformation from a traditional approach to agile development at RetailCo, three Scrum Masters from three different teams notice that a large amount of different methods is used throughout the organization.

Context
Knowledge sharing is only applied within teams, but not among several teams.

Problem
C-05: Facilitating Shared Context and Knowledge
C-19: Dealing with Internal Silos
C-39: Establishing a Culture of Continuous Improvement

Forces
People do not actively share their knowledge unless they are asked to. Without the possibility to openly discuss their practices, teams will not share their experiences and educate themselves about different alternatives. An agile organization can only improve if the teams try out things and do not stick to the same procedures. Many practices propose knowledge sharing within teams, but not across several teams.

Solution
Set up a Community of Practice for a specific domain.
A Community of Practice is a group of people 'who share a concern, a set of problems, or a passion about a topic' [83]. Participation is not limited to a set of people but is open to everyone in the organization, who is interested in the topic [63]. The intention is to enable frequent knowledge and expertise sharing between the participant [83]. The focus is to talk about practices that are applied and not to discuss theories. The participants of a Community of Practice are typically not from the same team but from many different teams all across the organization [83]. In the best case, many different practices can be

presented and discussed, leading to a wide knowledge base. Even though participating in a Community of Practice is voluntary, great numbers of participation can be reached if the participants feel the benefit in their work. Therefore, a Community of Practice should always have an interesting topic and a proper agenda, which is to be sent out with the invitations [63]. In addition, each Community of Practice should be lead by an expert, who is passionate to make the event a success and keep it on a frequent level [63]. Finally, set up an intranet page, where all information regarding the Community (e.g., agendas, or developed artifacts) are stored. This should be available for the whole organization [63].

Variants
A Community of Practice can be set up for a variety of domains. In practice, we identified Communities for the following domains: Architecture, Testing, Interfaces, Deployments, Leadership, Infrastructure.

Consequences
Benefits:

- Knowledge sharing is enabled.
- Silos are decreased as people from different parts of the organization meet.
- People feel that they are allowed to try out new things. This opens the opportunity to improve.
- The participants help each other to solve problems.
- The transformation is supported by a group of experienced and interested members of the organization.
- A broad knowledge base can lead to a competitive advantage [83].

Liabilities:

- People need to be allowed to take the time.
- If too few people participate, a Community of Practice does not fulfill its purpose.

See Also

- M-03: EMPOWERED COMMUNITY OF PRACTICE

Other Standards
SAFe®

Known Uses

- Electronics GmbH

- Global Insurance Corp
- LuxCarsCorp
- Retail Corp
- Software Inc

5.3.2. Publish Good Practices

Principle Overview	
ID	P-01
Name	PUBLISH GOOD PRACTICE
Alias	
Summary	To enable a culture of open communication and continuous improvement, people need to talk about what went well. Therefore, always publish good practices.
Type	Communication
Binding Nature	Recommended

Example
A Scrum Master at RetailCo successfully applied the method 'Supervision' within his team. However, there is no community of sharing good practices.

Context
There are not communication channels to update the whole organization about good practices. In addition, there is no culture of communication best practices.

Problem
C-4: Dealing with doubts in people about changes
C-5: Facilitating shared context and knowledge
C-33: Building trust of stakeholders in agile practices
C-39: Establishing a culture of continuous improvement
C-59: Establishing a common understanding of agile thinking and practices
C-91: Demonstrate Value Add of Agile Methods
C-94: Understanding the demand for becoming agile
C-110: Establishing an Agile Mindset

Forces
People are not conscious about the importance to talk about things that work well. By not sharing good practices, teams do not try out new things and keep applying the same methods unless told otherwise, consequently preventing improvement.

Consequences
Benefits:

- A culture of continuous learning is established.
- Open communication is facilitated.
- People get informed about good practices within and outside their organization.

- People are engaged to try out new things.

Liabilities:

- Open communication can lead to a higher tolerance to make mistakes and trying out things that are not suitable for the organization or the team.

See Also

- *CO-05: Agile Hour (Pattern Candidate)*
- *CO-06: Best Practices Webinar (Pattern Candidate)*
- V-02: GOOD PRACTICE NEWSLETTER

Known Uses

- Agile Consultants GmbH
- Global Incurance Corp
- IT Consultancy GmbH
- LuxCars Corp
- Retail Corp

5.3.3. Global Impediment Process

Pattern Overview	
ID	M-06
Name	GLOBAL IMPEDIMENT PROCESS
Alias	
Summary	This process describes how to identify, document and solve global impediments, after the team fails to solve them on their own.

Example
A team complained that they cannot involve the customer at early development stages due to a restriction that prohibits customer access to the development environment. Even though the Scrum Master is responsible for solving impediments, he does not know how to solve this one.

Context
The organization is currently transforming from traditional to agile approaches or has successfully transformed.

Problem
C-39: Establishing a Culture of Continuous Improvement
C-67: Encouraging Development Teams to Talk About Tasks and Impediments

Forces
There are obstructions within the development process that require action.
Whenever people notice their impediments are not being targeted, they lose motivation to talk about problems. Instead of resolving an impediment, teams find workarounds or create isolated applications.
An agile team constantly tries to improve its process.

Solution
Implement a Global Impediment Process to tackle impediments that teams cannot solve on their own. An impediment of a team will be included within the Global Impediment Process if no one else can solve it. The process is structured according to Figure 5.11.
The process includes a *Global Impediment Working Group*, which consists of people, who know the company well, therefore knowing the right people to solve an impediment. The Global Impediment Working Group meets every two weeks and discusses and prioritized new impediments. They add them to the Global Impediment Board and try to solve the impediments. Because of their knowledge about the company, the probability that they know people, who can solve the impediment, is very high.

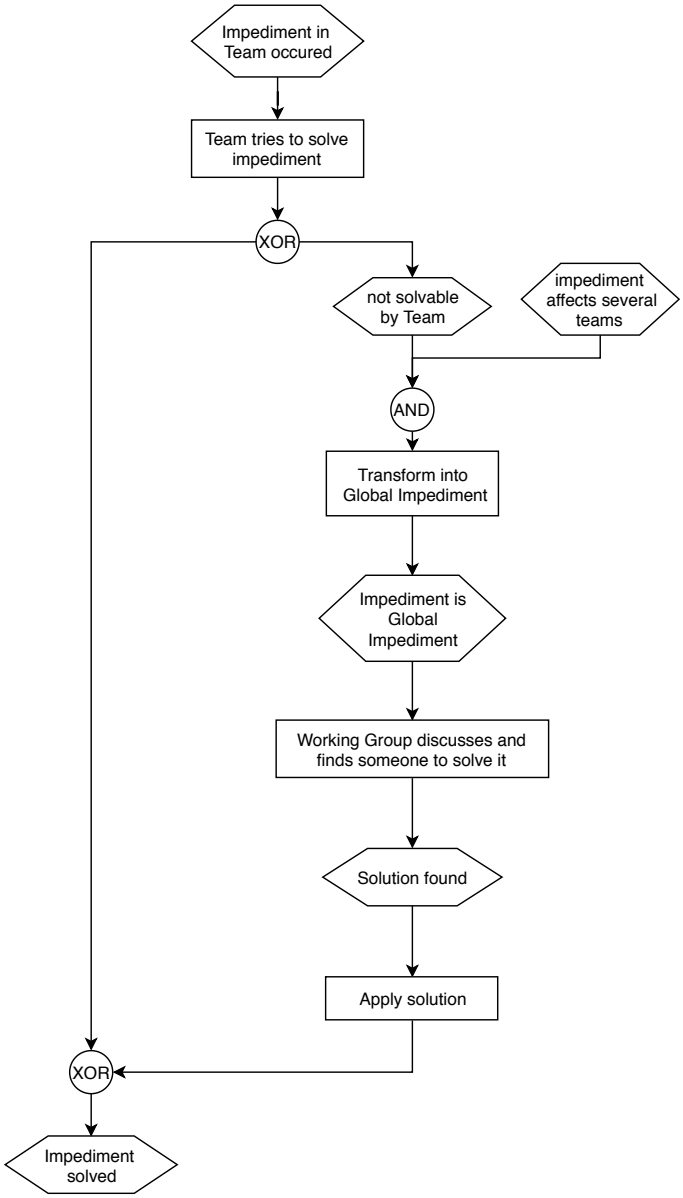


Figure 5.11.: Event-Driven Process Chain for the Global Impediment Process

Variants

1. Each impediment has an 'Owner', who is someone from the Working Group who also knows about the difficulty of the impediment.
2. Each impediment has a 'Supporter', who is someone who as a major influence on the solution of the impediment.
3. The submission of an impediment requires the A3 format [74].

Consequences
Benefits:

- Impediments get solved.
- Prioritization enables calculation of real cost caused by an impediment. This may increase resolution speed.
- The process enables transparency.
- The process minimized local applications and workarounds.
- By resolved impediments, development is more efficient.

Liabilities:

- The process requires increased effort for the participants of the Global Impediment Working Group.

See Also

- V-01: GLOBAL IMPEDIMENT BOARD

Known Uses

- Autonomous Cars Group
- Electronics GmbH
- IT Business Consultancy
- Retail Corp
- Software Inc

5.3.4. Global Impediment Board

Pattern Overview	
ID	V-01
Name	GLOBAL IMPEDIMENT BOARD
Alias	Global Impediment Backlog
Summary	This board displays all impediments of an organization, which could not be solved by teams themselves.
Type	Board

Example
RetailCo has established a Global Impediment Process to handle impediments that teams cannot solve on their own. However, the submitted impediments are only stored on a shared cloud drive. There is no way to detect the status of an impediment without opening its document.

Context
The organization has implemented a Global Impediment Process. The applied solutions are not documented, resulting in loss of knowledge.

Problem
C-67: Encouraging development teams to talk about tasks and impediments

Forces
Global Impediments need to be managed and tracked.

Solution
Set up a global impediment board to manage all global impediments throughout the Global Impediment Process. In large-scale agile development, a list with the following structure is frequently used:

Global Impediment Board

ID	Prioritization	Name	Handed in by	Description	Date	Owner	A3	Status
...	...	...	...	...	...	...	...	...
100	3	Dev Access	Team A	Customer cannot access development environment due to security guidelines	06/05/2019	John Doe	Link to A3	Ongoing
...	...	...	...	...	...	...	...	...

Figure 5.12.: Global Impediment Board

The ID is a consecutive, unique integer value that is used to identify an impediment. The prioritization is done by the Global Impediment Working Group and indicates the urgency of the impediment. Impediments with higher prioritization should be solved first. 'Handed in by' refers to the team or individual who handed in the impediment. The owner is someone from the Working Group, who is responsible for solving the impediment. The 'A3'-attribute is optional if the Global Impediment Process requires the submission of an impediment in the A3 format.

Variants
Depending on the organization, different attributes can be added or removed.

Consequences
Benefits:

- Everyone can view current global impediments and see if anyone else has a similar problem.
- Prioritization enables faster solving of emerging impediments that have a large impact on a team's organization.

Liabilities:

- It requires effort to manage the board.

See Also

- M-06: GLOBAL IMPEDIMENT PROCESS

Data Collection
The board can be digitally administrated in any digital collaboration tool by the Global Impediment Working Group. It is updated whenever an impediment occurs by a member of the Working Group. Only these members have writing permissions. It depends on the organization if the board should be public or kept private to the Working Group. The data model for the Global Impediment Board can be found in Figure 5.13.

Known Uses

- Autonomous Cars Group
- IT Business Consultancy
- Retail Corp
- Software Inc

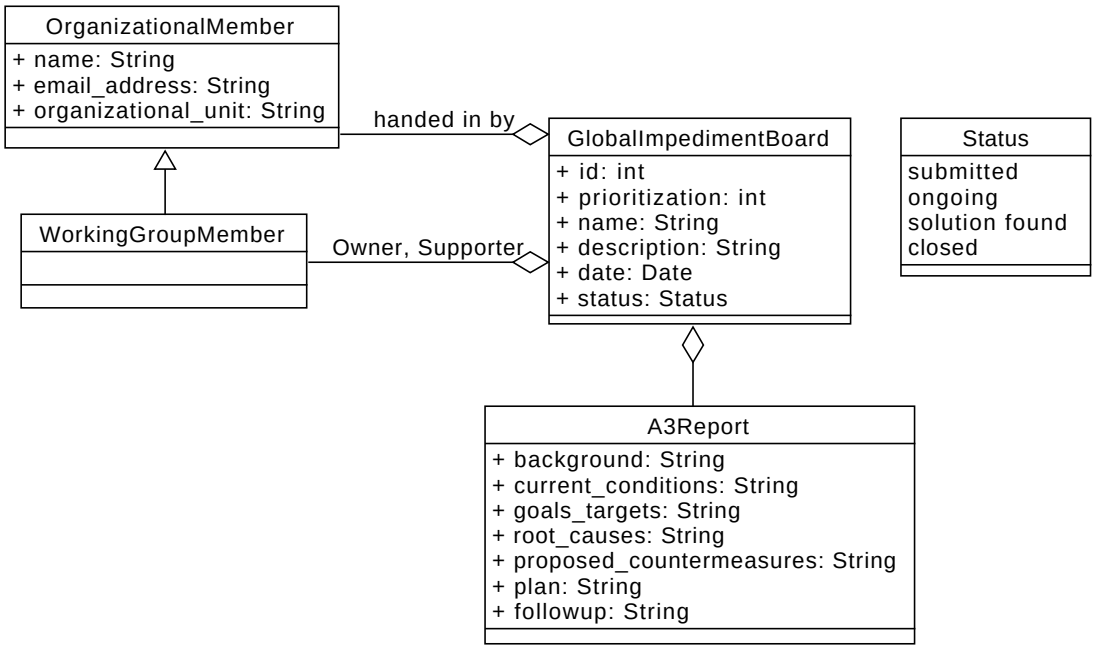


Figure 5.13.: Data Model for the V-Pattern Global Impediment Board

5.3.5. Don't use Frameworks as Recipes

Anti-Pattern Overview	
ID	A-01
Name	DON'T USE FRAMEWORKS AS RECIPES
Alias	
Summary	Instead of adopting a large-scale agile framework one-to-one and not training people in agile values and principles, the organization has to tailor a framework to their specific circumstances and train everyone to establish an agile mindset.

Example
The managing board of SoftwareInc hat decided to adopt agile methods on all hierarchy levels. They decide to implement the SAFe® framework one-to-one.

Context
In order to guide the agile transformation, the organization adopts a scaled agile framework.

Problem
C-7: Dealing with incorrect practices of agile development

Forces
It is easy to introduce methods, but it is hard to change people's minds.
Each enterprise has its own unique processes, which are not modeled in frameworks.

General Form
All methods, events and roles are clearly defined and a framework as been adopted one-to-one. However, there is no training on agile principles and agile values as this is not part of the framework. The new agile organization is not as efficient as expected. Many people are unsatisfied with the situation.

Consequences
Benefits:

- All agile practices of a framework are adopted.

Liabilities:

- People do not have an agile mindset.
- The organization is pseudo agile.

- The practices are not tailored to the organization's structure, and consequently, the organization cannot be as productive as possible.
- The organization spends a large amount of money to train people in role they do not need.

Revised Solution
Don't adopt an agile framework one-to-one. Always analyze which practices are relevant to the organization and which are not. Start a small-scale pilot first, and scale it to the whole organization after a successful pilot. Constantly inspect the organization and react to inefficiency accordingly.
Additionally, teach the organization to not only apply agile methods but to act and work according to agile values and principles. This requires extensive training and continuous review and improvement. Values and Principles are the basis for an efficient and value-creating organization.
In General, the use of an agile adoption framework is recommended [73, 77], which guides an organization through adopting and implementing agile practices.

See Also

- *P-11: Separate Observation from Solution (Pattern Candidate)*

6. Discussion

This chapter will evaluate, discuss, and reflect the results presented before (see Section 6.1) and point out this work's limitations (see Section 6.2.

6.1. Key Findings

The role of Agile Coaches and Scrum Masters in large-scale agile development

Section 2.3 describes the responsibilities and challenges of Agile Coaches and Scrum Masters. While it is easy to identify the tasks and scope of a Scrum Master, the literature on Agile Coaching is very limited. As the tasks mostly overlap, the identified concerns of both stakeholders have been merged. This is also justified by the results of the survey regarding recurring concerns. All respondents were Agile Coaches, but also marked Scrum Master-related concerns as relevant to them. This leads to the conclusion that Agile Coaches and Scrum Masters share most responsibilities and challenges. AC6 from the second series of interviews further confirmed this hypothesis by stating that he cannot answer the question if he is an Agile Coach or a Scrum Master because it is the same to him. In contrast, AC10 argues that he would tell us different concerns and patterns based on his viewpoint. He stated that he is both, and Agile Coach and a Scrum Master and is indeed able to differentiate between those two roles.
Further, two types of Agile Coaches have been observed during the interviews: (1) mindset-focused coaches and (2) process-focused coaches. The mindset-focused coaches often explained concerns regarding teamwork, agile mindsets, and communication. This lead to many patterns containing visualizations and workshops, e.g., Publish Good Practice, Celebrate Every Success, and Supervision. On the other side, the process-focused coaches explained methods and saw the agile transformation on a broader scope. They led to patterns like Objectives and Key Results, Piloting, and the Global Impediment Process. These observations fit the concept of Kelly [47] presented in section 2.3: the non-directive coaching approach is mostly used by the mindset-focused coaches, whereas the process-focused coaches often use the directive approach.

Recurring concerns in large-scale agile development

The interview is designed such that the participants do not know the existing concerns in advance. Therefore, we expected many duplicates between the existing concerns and the new ones mentioned during the interviews. However, it turns out that the literature

review [78] is missing some critical concerns of Agile Coaches and Scrum Masters, e.g., the challenge of managing external developers, emotional consequences triggered by change, or organizational transformation on all scaling levels. While [78] only identified concerns for Agile Coaches dealing with mindset and training, the interviews revealed that Agile Coaches also handle a large part of designing and executing the agile transformation of an organization. However, all concerns identified by Uludağ et al. [78] have been confirmed by this work.
Even though every respondent agreed that there are significant issues with wrong and difficult mindsets within the organization, only AC12 explained 'Establishing an Agile Mindset' from the ground as the top concern. As we expected this concern to be mentioned at least once during each interview, this result is surprising. In contrast, most Agile Coaches explained ways to create and strengthen agile mindsets, e.g., 'Acknowledging the Demand for Change', 'Understanding the Demand for Becoming Agile', or 'Demonstrate Value Add of Agile Methods'.

Patterns as a mean to document best practices

The decision to document best practices in the form of patterns received an enormous amount of positive feedback as elaborated in Section 4.2. Two Agile Coaches from the first interview series already promised to share their knowledge in the second interview series, as they liked the concept and wanted to participate. Other coaches also expressed their appreciation of our work during the interviews and are highly interested in the final artifact. Eventually, AC11 explained, he is thinking about using the pattern language for upcoming projects.
The distinction between three types of patterns, namely Methodology, Coordination, and Viewpoint, raised some initial discussions, which is also reflected in the evaluation results. Nevertheless, the interviews on identifying best practices showed that this distinction works well. It was easy to sort a pattern into one of the categories without the demand to discuss if it would fit into another category. This validates the usability of the Large-Scale Agile Development Pattern Language. In contrast, the distinction between a Principle and an Anti-Pattern was harder, as an Anti-Pattern can most of the time also be written as a Principle, e.g., the Anti-Pattern 'Don't Outsource Development' can be formulated as the Principle 'More Personal Contribution, Less External Labour'. This example has been taken from the interview with AC2 during the second interview series.

Best practices for addressing recurring concerns

The interviews on identifying best practices in large-scale agile development lead to several key findings. Brown et al. [12] stated, Anti-Patterns are 'a much more powerful and effective form for describing recurring solutions than design patterns'. Our research can confirm this. Even though we identified more patterns and principles than anti-patterns, many interviewees confirmed that describing what not to do is easier than describing what

to do when facing a problem. It is a difficult task to guide the interviewees into a direction, so they talk about best practices instead of talking about wrong practices.
Another aspect to be considered is the number of visualizations used. AC3 already stated this during the evaluation interview, and several Agile Coaches confirmed it during the second interview series: visualizations only provide value if they are actively used. Someone has to keep them updated, and the team has to use them, e.g., during a Retrospective. A large amount of outstanding visualizations does not provide any value to the team or organization if they are not actively used. Therefore, only a few, good visualizations should be considered. A problem detected for visualizations with the type *Board*, e.g., DeMo Board, KUDOS Board or Agile Goal Board, is the missing scaling factor. All of these boards were hanging at the walls within offices. Consequently, they are not applicable for distributed teams, which raises the demand for a digital alternative for all of these boards. However, many boards already were managed digitally, e.g., Global Impediment Board, Skill Matrix, and the Release Burn-up Chart.

6.2. Limitations

Semi-structured interviews serve as the main data source for this work. As part of a framework for developing semi-structured interviews, Kallio et al. [46] suggest to use three criteria for assessing the trustworthiness of a study: credibility, dependability and transferability [37].

Credibility. The credibility, also called *internal validity*, assesses whether a study measures what it is intended to measure [72]. To ensure high credibility of this study, interviewees were provided preparation material, describing the goal of the interview as well as the current research status. Each interview began with a short summary of the LSADPL, assuring that participants are aware of the large-scale agile context. By letting the interviewees list their personal top three concerns in the beginning without showing them the existing concerns, we could identify even more concerns that literature could [78]. However, due to the limited number of interviews, it is possible that there are still concerns and pattern(s) (candidates) not included in our results. As the interviews were also limited to 90 minutes, we could not capture every concept an interviewee uses.

Dependability. The dependability, or *Reliability* of a study requires the same results when the study is repeated within 'the same context, with the same methods and with the same participants' [72]. In general, each participant has his own set of tools he frequently uses. However, depending on the circumstances, an Agile Coach can present different patterns than during this study. The probability that he mentions a method he frequently used during the time this study has been conducted, is very high. In contrast, conducting this study a year later, this method could not be in use anymore, e.g., because the organization is in a different state of the agile transformation and requires different methods. Therefore, the

same results are not guaranteed.

Transferability. The *external validity* describes to which degree the findings can be generalized to a broader population [72]. As the nature of patterns is to be proven [17], the results can be generalized for the stakeholders 'Agile Coach' and 'Scrum Master'. However, the external validity can be greatly increased by applying the patterns in practice, observing the context, implementation, and consequences and adapt the findings.

Eventually, [37] presents a fourth criterion for qualitative research, which is called **Confirmability**. This term refers to the *objectivity* of the study. The results must not be biased by the researcher's 'characteristics and preferences' [72] instead must be the result of the interviewees' input [72]. By sticking to the interview guide (see Appendix A.2), the bias by the researchers has been minimized.

Additional limitations arise from the Pattern-based Research Design, which has only been partly conducted (see Figure 1.2). A validation of the identified patterns in practice has not been conducted. Therefore, the confidence of the patterns greatly differs based on the amount of information provided by the interviewees. For example, we were only shown one Global Impediment Board, whereas other participants only confirmed that they have such a board without showing it to us. It is not validated whether the patterns are written in a way that allows easy implementation in practice.

7. Conclusion

This chapter summarizes all research results in Section 7.1 and gives an outlook on future research projects in Section 7.2.

7.1. Summary

This thesis identified recurring concerns of Scrum Masters and Agile Coaches in the context of large-scale agile development and determined best practices for encountering these challenges. The best practices were documented in the pattern form as part of a large-scale agile development pattern language. In the first step, the large-scale agile development pattern language has been developed based on [78] and [32]. The initial version contained the elements stakeholder, initiative, challenge, principle, coordination pattern, viewpoint pattern, methodology pattern, and anti-pattern. A stakeholder is everyone who as an interest in a large-scale agile development undertaking, while an initiative is a group of stakeholders. Challenges hinder stakeholders in performing tasks and patterns are solutions to solve those challenges. A solution can either be a meeting (CO-Pattern), a process or method (M-Pattern), or a visualization or metric (V-Pattern). Further, principles are general rules that stakeholders need to stick to, which can also be measured by V-Patterns, e.g., by a KPI. Finally, an Anti-Pattern describes a solution with a large number of great consequences and presents a revised solution to be applied instead. An evaluation based on 14 structured interviews lead to six adjustments within the language, involving the removal of the element 'initiative' as well as the connection of anti-patterns with all other elements. The element 'challenge' has been renamed to 'concern', adhering to [45] and receiving a broader context. The implementation section has been removed from every pattern type and the element 'principle' received a field 'Binding Nature', but lost its solution section. The final version of the pattern language is described in Section 4.3. The second part of this thesis consists of 13 semi-structured interviews conducted for the identification of recurring concerns and best practices. We interviewed Agile Coaches and Scrum Masters from eleven different companies, with experience in large-scale agile development of at least one year. Overall, we documented 36 concerns, which are explained in detail in Section 5.2.1. This number decreased to 23 after removing duplicates and merging concerns with similar scope. The complete list of final concerns can be found in Appendix B.1. All concerns identified by Uludağ et al. [78] have been confirmed as *recurring* as well as all 23 concerns identified by this thesis. Finally, we observed 76 possible best practices, also called pattern candidates. This divides into 21 M-Pattern candidates, 18 V-Pattern candidates,

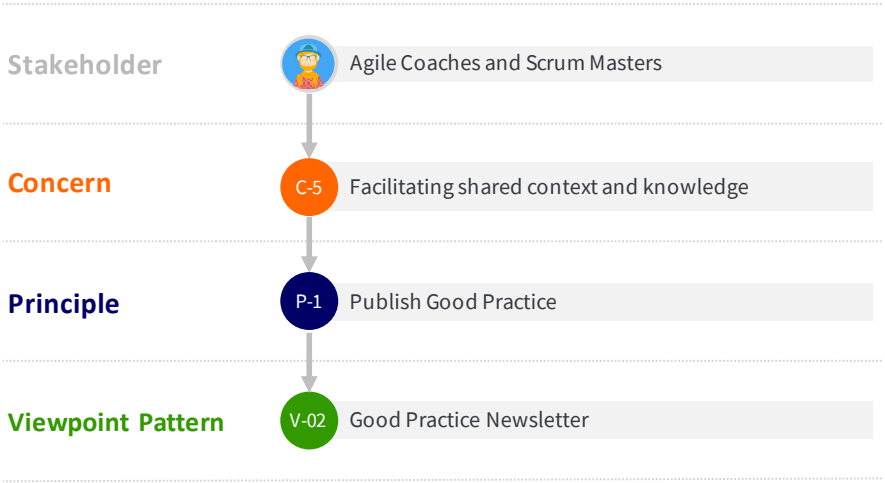


Figure 7.1.: Possible Use Case for the Large-Scale Agile Development Pattern Language

14 CO-Pattern candidates, 12 Principle candidates, and 10 Anti-Pattern candidates. After applying the rule of three [17], the following patterns remained: Supervision, Community of Practice, Empowered Community of Practice, Global Impediment Process, Role Focus, Objectives and Key Results, Global Impediment Board, Good Practice Newsletter, Piloting, Publish Good Practice, Consensus-Based Decisions, Celebrate Every Success, Explain Meeting Purpose, Don't Use Frameworks as Recipes, and Don't Overshoot Coordination Meetings. A possible way to use the LSADPL is shown in Figure 7.1. When viewing the stakeholder 'Agile Coach' or 'Scrum Master' all attached concerns should be displayed. One of those concerns is called 'Facilitating shared context and knowledge', which leads to the principle 'Publish Good Practice'. A link to the V-Pattern 'Good Practice Newsletter' within the principles shows the user ways to adhere to this principle. Finally, the Agile Coach has been shown a proven, working way to facilitate shared context and knowledge. The main benefit of the LSADPL is the high degree of usability due to five different pattern categories as well as structural mechanisms, like stakeholders and concerns. The user is able to search for suitable patterns much faster, compared to other pattern languages (see Section 3.2).

7.2. Future Work

This thesis sets the foundation for further research on recurring concerns and best practice of other stakeholders with the aim to complete the LSADPL. It would be of high value to observe other stakeholders, e.g., Product Owner, Developer, Architect, or Manager. Not only did many Agile Coaches explained issues with the Higher Management, but 'Higher Management Interferences' has also been found in the literature. Hence, mid- and high-level managers should be considered as additional stakeholders to be interviewed. An

important aspect to incorporate when interviewing other stakeholders are the existing pattern candidates. It is still possible that other stakeholders also apply solutions that were explained by Agile Coaches and Scrum Masters. Therefore, all pattern candidates can still become a pattern throughout future research. Due to the limited time frame of this thesis, further studying the application of the patterns in practice could not be conducted. However, those studies would provide valuable insights into the quality of the patterns. Based on such studies, the patterns could be extended with even more relevant information. Not only positive and negative consequences would be directly observable, but the solution section could be extended with steps on how to apply the pattern in practice (see Figure 1.2).

A. Appendix

The following chapter presents the questionnaires used in the interviews on evaluating the large-scale agile development pattern language (Section A.1) as well as the questionnaire for identifying recurring concerns and best practices (Section A.2).

A.1. Evaluation Interviews

1. **General Questions**
   a) How is your role denoted within your company?
   b) Since how many years are you working in the area of large-scale agile development?
   c) Since how many years is your company conducting large-scale agile development?
   d) Is your company acting on national or international level?
   e) In which country is your company's headquarter located?
   f) Which sector does your company belong to?
   g) How many people does your company employ?
2. **Questions regarding Elements of the Large-Scale Agile Development Pattern Language**
   a) Stakeholder
      i. In your opinion, how valuable is the inclusion of the element "Stakeholder" into the concept model of the LSADPL?
      ii. Why would the element "Stakeholder" be valuable/not valuable to you?
   b) Initiative
      i. In your opinion, how valuable is the inclusion of the element "Initiative" into the concept model of the LSADPL?
      ii. Why would the element "Initiative" be valuable/not valuable to you?
   c) Challenge
      i. In your opinion, how valuable is the inclusion of the element "Challenge" into the concept model of the LSADPL?

ii. Why would the element "Challenge" be valuable/not valuable to you?

d) Principle

i. In your opinion, how valuable is the inclusion of the element "Principle" into the concept model of the LSADPL?

ii. Why would the element "Principle" be valuable/not valuable to you?

e) Coordination Pattern and Methodology Pattern

i. In your opinion, how valuable is the inclusion of the element "Coordination Pattern" into the concept model of the LSADPL?

ii. Why would the element "Coordination Pattern" be valuable/not valuable to you?

iii. In your opinion, how valuable is the inclusion of the element "Methodology Pattern" into the concept model of the LSADPL?

iv. Why would the element "Methodology Pattern" be valuable/not valuable to you?

v. Would the differentiation between Coordination Pattern and Methodology Pattern help you?

vi. Why would this differentiation be helpful/not helpful to you?

f) Viewpoint Pattern

i. In your opinion, how valuable is the inclusion of the element "Viewpoint Pattern" into the concept model of the LSADPL?

ii. Why would the element "Viewpoint Pattern" be valuable/not valuable to you?

g) Anti-Pattern

i. In your opinion, how valuable is the inclusion of the element "Anti-Pattern" into the concept model of the LSADPL?

ii. Why would the element "Anti-Pattern" be valuable/not valuable to you?

iii. In your opinion, should the element "Anti-Pattern" be connected with another element of the concept model?

iv. If yes, with which element of the concept model should the element "Anti-Pattern" be connected and why?

3. **Questions regarding the relations within the Large-Scale Agile Development Pattern Language**

a) In your opinion, how meaningful is the relation between the elements "Stakeholder" and "Challenge"?

b) In your opinion, how meaningful is the relation between the elements "Initiative" and "Challenge"?

c) In your opinion, how meaningful is the relation between the elements "Challenge" and "Principle"?

d) In your opinion, how meaningful is the relation between the elements "Challenge" and "Coordination Pattern"?

e) In your opinion, how meaningful is the relation between the elements "Challenge" and "Methodology Pattern"?

f) In your opinion, how meaningful is the relation between the elements "Challenge" and "Viewpoint Pattern"?

g) In your opinion, how meaningful is the relation between the elements "Principle" and "Viewpoint Pattern"?

h) In your opinion, how meaningful is the relation between the elements "Coordination Pattern" and "Viewpoint Pattern"?

i) In your opinion, how meaningful is the relation between the elements "Methodology Pattern" and "Viewpoint Pattern"?

j) In your opinion, how meaningful is the relation between the elements "Coordination Pattern" and "Methodology Pattern"?

4. **Discussion**

a) Is there any other concept, that you would like to have included in the Large-Scale Agile Development Pattern Language?

b) Your comments:

c) Would you use patterns for addressing recurring challenges in scaled agile development?

d) Would patterns regarding scaled agile development help you in your job?

A.2. Interviews on Identification of Recurring Concerns and Patterns

1. **General Questions**

a) How is your role denoted within your company?

b) Would you call yourself an Agile Coach or a Scrum Master?

c) Since how many years are you working in the area of large-scale agile development?

d) Since how many years is your company conducting large-scale agile development?

e) Is your company acting on national or international level?

f) Which sector does your company belong to?

g) How many people does your company employ?

h) May we contact you again as part of the research project? (For example, to check the finished patterns, or to ask further questions about your answers.)

2. **Questions on Recurring Concerns**

   a) What are the recurring concerns you face in your role as Scrum Master/Agile Coach? Please name your top three.

   b) On which scaling level do these concerns typically arise? You can name more than one.

   c) In which category would you classify this concern? You can name more than one. If none fits, you can create a new one.

3. **Questions on Best Practices for Top Three Concerns**

   a) What do you do to encounter this concern?

   b) What should one never do to encounter this concern?

4. **Questions on Concerns identified in Literature**

   a) Which of these concerns identified in literature do you face in your role as Agile Coach/Scrum Master?

   b) What do you do to encounter this concern?

   c) What should one never do to encounter this concern?

5. **Questions for documenting Patterns and Principles**

   a) Is there a specific project or product in which this pattern was used?

   b) Which problems should be addressed by this pattern?

   c) Under which circumstances does this problem occur?

   d) Why is this problem hard to solve?

   e) How do you solve the problem?

   f) Which variants do exist?

   g) What are the benefits of this solution?

   h) What are the liabilities of this solution?

   i) What other standards or frameworks recommend this pattern?

j) *Only for V-Patterns:* Were does the data for this V-Pattern come from? Were are they stored and who administers them?

6. **Questions for documenting Anti-Patterns**
   a) Is there a specific project or product in which this anti-pattern was used?
   b) Which problems should be addressed by this anti-pattern?
   c) Under which circumstances does this problem occur?
   d) Why is this problem hard to solve?
   e) How can this anti-pattern be found in practice?
   f) What are the benefits of this solution?
   g) What are the liabilities of this solution?
   h) How should this anti-pattern be avoided?
7. **Discussion**
   a) Do you have any further comments?

B. Appendix

B.1. Documentation of New Concerns

The following table summarizes all recurring concerns in large-scale agile development identified within thirteen interviews with Agile Coaches and Scrum Master.

ID	Name	Description	Category	Scaling Level	Source
C-81	Enable Change from Process to Product Orientation	The value stream within an organization needs to be altered from a process-oriented to a product-oriented view.The organization needs to get a rather customer-centered value stream. It is the task of the Agile Coach to trigger and guide this change.	Methodology	Enterprise	AC1
C-83	Dealing with External Developers	Because of cost reduction and missing know-how, development is often outsourced to external providers. However, this brings up several challenges especially if the project management is still internal. Along with many communication issues, the internal and external team members need a clear model to work together, especially if they are distributed. Moreover, legal requirements increase the complexity when it comes to assigning work and paying the external developers.	Communication & Coordination	IT Organization	AC2
C-85	Self-Awareness as an Agile Coach	During the agile transformation, many emotions are triggered in people's minds. Therefore, it is important that the Agile Coach has a high degree of authenticity and know himself well. He needs to insist on his opinions and viewpoints. Therefore, he has to understand what kind of a person he is and how others notice him. The Agile Coach should not mimic something he has seen somewhere else but be himself instead.	Culture & Mindset	All	AC3

ID	Name	Description	Category	Scaling Level	Source
C-86	Emotional Consequences of Agile Transformations	People need to understand that change brings emotional consequences that they cannot suppress because of natural instincts. The Agile Coach has to make it clear that emotions are fine and a natural thing that affects everyone. It is important that the Agile Coach does not miss this human factor during the agile transformation.	Culture & Mindset	All	AC3
C-87	Patience During the Agile Transformation	Agile transformations take time and re-quire much patience. The Agile Coach needs to constantly remind the organization that it takes time to change not only process but also mindsets. He needs to make clear that change is a continuum and things do not change overnight.	Culture & Mindset	All	AC3
C-88	Building an Agile Organization around Norms and Standards	Each team uses different technologies and practices. An agile organization needs to cope with this heterogeneity and build up a shared knowledge base. Standards and norms not only need to be defined but also elaborated and implemented.	Tooling	Program	AC4

ID	Name	Description	Category	Scaling Level	Source
C-89	Establishing Equality among Cross-Functional Teams	In cross-functional teams each team member typically has a wide skill set and is able to fulfill many different tasks. Consequently, if a team member calls in sick, another team member takes over his work, because he already has the required skills and knowledge. Still, each team member has his specific field of expertise, in which he is outstanding. This turns into a negative concern if this team member always gets the tasks focusing on his specialization and not the team member with free capacity. It gets even worse if this applies not only to one team member but to the whole team, dismissing the concept of a cross-functional team.	Project Management	Team	AC4
C-90	Objective Measuring Methods	Every questionnaire is biased to some degree, consequently being very subjective. Depending on the parameters included, KPIs always lead to a certain direction the creator wants to achieve.There is a demand to monitor the status of an objective view.	Tooling	Team	AC5
C-91	Demonstrate Value Add of Agile Methods	tightly relates to C-46'Dealing with closed-mindedness' as people, who oppose agile methods, first need to understand their value add. They need to internalize why agile methods are important for the success of the organization instead of only understanding how these methods are applied. Therefore, it is the Agile Coach's responsibility to explain and practically demonstrate the value add of agile methods.	Culture & Mindset	All	AC6

ID	Name	Description	Category	Scaling Level	Source
C-92	Coordination of Multi-Vendor Teams	In contrast to C-83 'Dealing with external developers', which is more generic and only deals with the fact that there are external developers within an agile team, this concern specializes on multiple external teams from different vendors. This brings more complexity to the product, more communication and coordination issues, and special legal challenges.	Communication & Coordination	Program	AC6
C-93	End-to-End Usable Functionality Implemented in One Iteration	Traditional project management life cycles start with an analysis of the requirements, followed by a design phase, and delivery. Agile teams often work with user stories, meaning they need to undergo all of these phases within one iteration. This gets even more complicated if different systems are affected.	Requirements Engineering	Program, Team	AC6
C-94	Understanding the Demand for Becoming Agile	Both, stakeholder and team members need to understand why the agile transformation is necessary and how it is done in general. Moreover, the Agile Coach has to explain when agile practices can be applied and under which circumstances agility does not make sense.	Culture & Mindset	All	AC7, AC11

ID	Name	Description	Category	Scaling Level	Source
C-95	Lacking Orientation because of Missing Leadership	Self-organization also requires a minimum degree of leadership. People are afraid during the agile transformation, because of much change. AC7 mentioned the example that a Software Architect loses his role during the introduction of Scrum and becomes only a part of the development team. A part of the leadership team should provide orientation to these people and mediate between 'stability and instability' (AC7). However, these mediators of-ten are external agile coaches and not internal managers. Therefore, the agile coach has to prepare the organization for the time after his retirement to prevent missing orientation.	Culture & Mindset	All	AC7
C-96	Decisions on Higher Levels reach Lower Levels	Decisions that have been made on a higher hierarchy level need to trickle down to every team member, which is hard to establish. Because all members of an organization need to contribute to a common goal, everyone needs to be informed about the strategy.	Communication & Coordination	All	AC7
C-97	Frameworks on All Hierarchy Levels	An organization needs to develop a frame-work that captures agility on all hierarchy levels. This framework can never be a standard framework (e.g., SAFe), mapped to the organization one on one. An Agile Coach or a team of Agile Coaches, in combination with internals, need to tailor frameworks such that they fit the organization's strategy and processes	Knowledge Management	All	AC7

ID	Name	Description	Category	Scaling Level	Source
C-100	Higher Soft Skill Requirements	Due to more communication with more people, soft skills are much more important in agile organizations. Agile Coaches need to find an effective training method to increase the soft skills of each team member.	Training	Team	AC8
C-102	Motivating Leadership to Talk to Teams	The Product Owner should not be the only one who talks to the team and communicates the higher-level decisions as well as the common product vision. The leadership also needs to know what is happening within the teams and how they are performing.	Culture & Mindset	Team	AC9
C-104	Definition of the Product Owner Role	Especially in large companies, ownership regarding parts of the product is very distributed. In the beginning, it is hard to59 5. Identification of Recurring Concerns and Best Practices identify who the Product Owner is and what responsibilities he has. It is difficult to understand who is able to take which decisions. Consequently, this concern does not target the definition of tasks and responsibilities of the Product Owner role. Rather, it to identify the Product Owner.	Project Management	Portfolio	AC10
C-106	Agile Program Management	A program typically has several milestones that need to be reached at a certain point in time. This is contrary to the agile approach. AC10explains that the point in time will definitely come, but it is not known in advance if the status of the program will be satisfactory. This aspect of agile program management needs to be accepted by the organization. However, managers fear this degree of uncertainty for a whole program.	Project Management	Program	AC10

ID	Name	Description	Category	Scaling Level	Source
C-109	Learning to Improve as an Organization	Not only individuals or teams must improve, but also the organization as a whole. They need to decide on things jointly and work on topics together.	Culture & Mindset	All	AC11, AC13
C-110	Establishing an Agile Mindset	This concern faces the general challenge on how to establish an agile mindset within an individual or a team.	Culture & Mindset	All	AC12
C-111	Coaching on Higher Management Level	part of the classic agile frameworks, like Scrum, but has to be considered in scaled agile methods. This concern not also addresses how to coach managers, but also how to achieve the mandate to coach managers	Culture & Mindset	Enterprise, IT Organization, Team	AC12, AC13
C-112	Identifying Dependencies between Teams on Cross-Domain Level	this concern refers to technical dependencies between teams on cross-domain level. The focus only lies on identifying and managing those dependencies.	Enterprise Architecture	Enterprise, IT Organization, Team	AC12

Table B.1.: Complete Documentation of new Concerns identified for Agile Coaches and Scrum Master

C. Appendix

C.1. Documentation of Patterns

C.1.1. Supervision

Pattern Overview	
ID	CO-01
Name	SUPERVISION
Alias	
Summary	While facing many different problems, a team can set up a Supervision. In this meeting, four to eight people meet for about three hours to list and prioritize all problems. Two problems are eventually chosen via voting and the participants collectively find and evaluate solutions to those problems.

Example
Five Agile Coaches at SoftwareInc each have their individual problems, which they cannot find solutions to.

Context
A set of people from the same organization have multiple problems and impediments, which they need to encounter.

Problem
C-74: Empowering Agile Teams to Make Decisions
C-67: Encouraging development teams to talk about tasks and impediments

Forces
People want to solve all problems at once.

Solution
Set up a supervision meeting with four to eight participants for at maximum three hours. A typical agenda is structured as followed:

1. **Casting**: Every participant thinks of one to two problems he wants to discuss, and every problem is shortly introduced afterward. In the end, a voting decides which

two problems are going to be discussed in the current Supervision.

2. **Telling**: The person who introduced the problem, called the storyteller, has to explain it in more detail. Nobody else is talking, just listening
3. **Asking**: Everyone can ask questions about the problem.
4. **Hypothesis**: Everyone but the storyteller is discussing the problem and state hypotheses about it. During this stage, the storyteller should physically away from the others, for example by leaving the room or staying behind a flip chart.
5. **Feedback**: The storyteller evaluates the hypotheses.
6. **Solution**: The team presents solutions to the problem.
7. **Feedback**: The Storyteller evaluates the solutions and explains which are feasible and which are not. He decides which solution will be applied in practice.

Variants
A Supervision can be done within one team, or on cross-team level with people from the same domain. Domain-specific Supervisions can focus on problems by Agile Coaches, Product Owner or Architects.

Consequences
Benefits:

- All problems are identified in a structured manner.
- By prioritizing the problems, the most important problems can be encountered first.
- Solutions to the problems are gathered by different people, resulting in a wider range of possible solutions with each different benefits and drawbacks.

Liabilities:

- Not all problems can be discussed.
- Participants might not feel valued if their problem is not discussed.
- Problems with low priority can have larger impact in the future.

Known Uses

- Electronics GmbH
- IT Business Consultancy
- Retail Corp

C.1.2. Empowered Community of Practice

Pattern Overview	
ID	M-03
Name	EMPOWERED COMMUNITY OF PRACTICE
Alias	*none*
Summary	In contrast to a normal Community of Practice, an Empowered Community of Practice is able to make domain-specific decisions for a whole organization.

Example
During the agile transformation at SoftwareInc, the steering committee notices the demand for a process for collaborative decision-making on standards, like tools and libraries to be used throughout the whole organization.

Context
An organization needs to make decisions that are mandatory for all.

Problem
C-88: Building an Organization around Norms and Standards

Forces
Top-down decisions are often made by managers, who are lacking operational experience. People affected by the decisions are not integrated into the decision-making process.

Solution
Set up an Empowered Community of Practice.
In contrast to a COMMUNITY OF PRACTICE, an Empowered Community is able to take binding decisions for the whole organization that act as standards. The participants are not from the same team, but from many different teams across the organization, working in the same domain. The goal is to include as many affected people as possible into the decision-making process. Figure C.1 illustrates the five process steps of an Empowered Community of Practice: at first, every participant can make suggestions on the standard to be defined. Afterward, the suggestions are discussed and elaborated. Some proposals may be discarded during this phase, the benefits, drawbacks, and the suitability for the organization of all proposals are analyzed. Finally, a voting decides on the proposal to be applied. For voting, the principle of CONSENSUS-BASED DECISIONS should be considered. After the decision, the standard is rolled-out in the organization. The whole process may take several meetings.

Figure C.1.: Process Steps during an Empowered Community of Practice

Variants
An Empowered Community of Practice can be set up for a variety of domains. In practice, we identified Communities for the following domains: Architecture, Testing, Interfaces, Deployments, Leadership, Infrastructure.

Consequences
Benefits:

- Top-down decisions are reduced.
- People affected by the decisions are included in the decision-making process.
- Decisions are based on a wide range of experience.
- Organization-wide standards are defined.

Liabilities:

- An Empowered Community of Practice can lead to long discussion.

See Also

- CO-02: COMMUNITY OF PRACTICE
- P-04: CONSENSUS-BASED DECISIONS
- *P-06: People who are affected by change must be involved in the execution of change (Pattern Candidate)*

Known Uses

- Autonomous Cars Group
- Global Insurance Corp
- Retail Corp

C.1.3. Role Focus

Pattern Overview	
ID	M-1
Name	Role Focus
Alias	
Summary	As many frameworks require, every employee in an organization must assume an agile role. For this, the organization first needs to define which roles they need, define their responsibilities and start a pilot.

Example
SoftwareInc wants to transform its currently traditional development process to an agile process.

Context
The organization needs to flatten hierarchies and create a focus on roles.

Problem
C-56: Defining Clear Roles and Responsibilities
C-81: Enable Change from Process to Product Orientation

Forces
Frameworks propose a set of roles to be implemented by an organization. However, not all roles are required.

Solution To ensure that the organization only includes the roles required, apply the following process:

1. Define your roles: which roles does the organization require? Which roles proposed by frameworks do we need and which additional roles are required by unique organizational characteristics?

2. Pilot the role concept and agile processes within two groups of 120 to 130 employees.

3. Learn and adapt from the pilot

4. Apply the concept to the whole organization.

Each team lead has to decide on one of the following focus: Personnel, process, product.
With a focus on 'Process', they become a Scrum Master or Agile Coach.
With a focus on 'Products', they become a Product Owner.
Product Owner and Scrum Master are part of Project Organization. The team members

become the Development Team.

Variants
Employees can also decide if they become Personnel Specialist. This role remains part of the Line Organization.

Consequences
Benefits:

- The organization can handle products more flexible.
- The organization has as few roles as necessary..
- The organization can react faster to change.
- The organization focuses more on products and value creation.
- Everyone focuses on their core competence.
- The organization is able to create better products.

Liabilities:

- Everyone wants to be Product Owner, because of this role's better reputation.
- This method may bring up new silos.
- This method requires a large coordination effort and a complete organizational change.
- Young employees might not fit into any role.

See Also

- *A-10: Don't include Pseudo Product Owner (Pattern Candidate)*
- P-05: PILOTING

Other Standards Known Uses

- Electronics GmbH
- LuxCars Corp
- Retail Corp
- Software Inc

C.1.4. Objectives and Key Results

Pattern Overview	
ID	M-02
Name	OBJECTIVES AND KEY RESULTS
Alias	
Summary	This method is a way to achieve that a whole organization works towards a common goal or vision. Qualitative goals (‚Objectives') are defined and measured by quantitative ‚Key Results'.

Example
The managing board of SoftwareInc demand for higher click rates on their website. However, communicating such a goal throughout multiple channels has not lead to fulfillment within the last times, because teams did not know how they could contribute to it.

Context
An agile organization needs to work towards common goals without a detailed road map.

Problem
C-90: Objectives Measuring Methods
C-96: Decisions on higher levels reach lower levels
C-42: Common Vision

Forces
There is a company strategy, but it is not reflected in the teams' actions. It is hard to track if the strategy is followed and if the organization is making progress towards a common direction.

Solution
Implement Objectives and Key Results.
Objectives and Key Results is a common goal management method. In the beginning, three qualitative objectives (the goals) are defined. The key results are measurable, quantitative results (KPIs) that lead to reaching the objective. The OKRs are defined and evaluated every quarter and do not define a detailed road map on how to achieve the key results. Key results track the status to reaching an objective in percentage. In practice it is almost impossible to reach 100% on all key results; therefore 60% to 70% is agreed to be a good value.

Variants
Top-Down Approach: The higher management can set top-level (or even company-level) OKRs, which are communicated within the company. Each team defines its own OKRs based on the top-level OKRs.

Bottom-Up Approach: Each team defines their own OKRs, which are aggregated on higher levels (e.g., Product OKRs, Domaine OKRs, ...).
The time frame for the definition of new OKRs can vary based on the size of the organization and the OKR hierarchy. Very large OKRs can be defined for a time frame of several years.

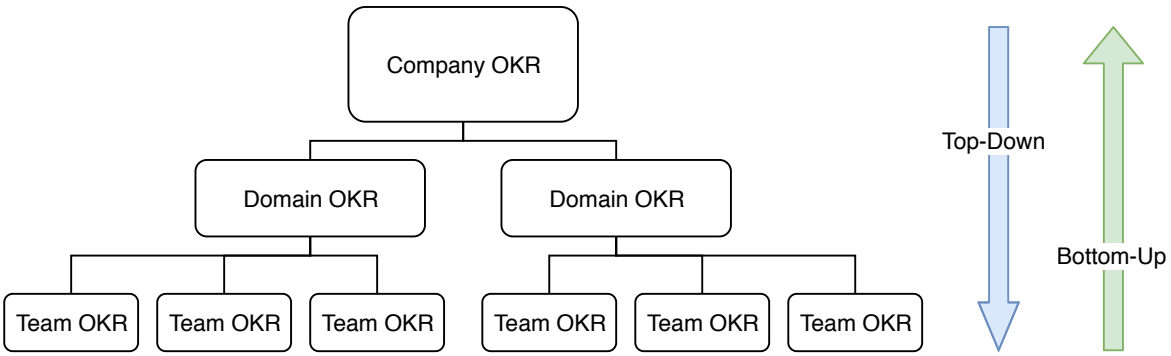


Figure C.2.: Bottom-Up and Town-Down Approach of Objectives and Key Results

Consequences
Benefits:

- The amount of Work in Progress is constrained.
- The work is outcome-centered.
- Self-organization is supported.
- The organization has a common vision that everyone works towards.

Liabilities:

- The Bottom-Up Approach consumes more time and needs increased coordination.

Known Uses

- Autonomous Cars Group
- LuxCars Corp
- Retail Corp

C.1.5. Piloting

Principle Overview	
ID	P-5
Name	PILOTING
Alias	
Summary	A pilot is a test project that is done before the agile methods are applied to a larger part of the organization [24]. Always run pilot projects, verify what works well, learn from mistakes, and adapt this experience when scaling the methods to a larger group [40].
Type	Communication
Binding Nature	Mandatory

Example
The managing board of SoftwareInc wants to transform the IT organization from the traditional to an agile approach. However, they are unsure, if it fits the organization and how to do this in general.

Context
An organization plans to transform their development from traditional to agile.

Problem
C-81: Enable change from process to product orientation

Forces
It is not possible to know in advance which methods and practices fit the organization best.

Variants
A pilot project can have a business sponsor, who communicates the pilot throughout the organization [56].

Consequences
Benefits:

- The organization can decide if they want to adopt large-scale agile methods or if they do not [40].
- A pilot is a first indicator for deciding which agile practices work well for the organization and which do not [26].
- Impediments and risks can be identified early [40].
- A successful pilot increases the acceptance of agile practices [26].

Liabilities:

- It takes time and manpower to conduct a pilot.
- If a pilot reveals many risks and impediments, doubts about an agile transformation increase [40].

Known Uses

- Electronics GmbH
- Global Insurance Corp
- LuxCars Corp
- Software Inc
- Retail Corp

C.1.6. Consensus-Based Decisions

Principle Overview	
ID	P-04
Name	CONSENSUS-BASED DECISIONS
Alias	
Summary	To make binding decisions, everyone has to agree to a proposal.
Type	Organizational
Binding Nature	Recommended

Example
During an Empowered Community of Practice the participants need to decide on a standard format for submitting global impediments. This decision affects the whole organization.

Context
Decisions need to be made for a whole organization.

Problem
C-88: Building an Agile Organization around Norms and Standards

Forces
There are many different suggestions, each having their own benefits and drawbacks. Everyone wants his suggestion to be realized.

Consequences
Benefits:

- The organization/team implements a solution everyone has agreed to.
- This type of decision-making avoids later conflicts.
- This principle facilitates standardization across large organizations.

Liabilities:

- Reaching consensus can lead to long discussions.

See Also

- M-03: EMPOWERED COMMUNITY OF PRACTICE

Known Uses

- Autonomous Cars Group

- Global Insurance Corp
- Retail Corp

C.1.7. Celebrate Every Success

Principle Overview	
ID	P-03
Name	CELEBRATE EVERY SUCCESS
Alias	
Summary	The team or the organization should celebrate and communicate every small and major success during the agile transformation. It does not matter how it is celebrated. The important aspect is to make success visible.
Type	Communication
Binding Nature	Recommended

Example
One year after the agile transformation of RetailCo has started, teams are wondering what has happened since then. Current working circumstances are a bit chaotic and it seems there has been no progress made in a while.

Context
People lack motivation to continue with the agile transformation, because they do not see progress.

Problem
C-87: Patience during the Agile Transformation

Forces
Transforming a large organization takes a lot of time and success is not always directly observable to everyone. Not seeing progress can be demotivating.

Consequences
Benefits:

- Celebration of small successes triggers positive emotions.
- The organization is more motivated to continue with the agile transformation.
- The organization is reminded that the agile transformation is an ongoing process that takes time.
- The organization recognizes its progress.
- People's work is appreciated and valued.

Liabilities:

- People, who are against the agile transformation, may feel annoyed by this.

See Also

- P-01: PUBLISH GOOD PRACTICE
- V-02: GOOD PRACTICE NEWSLETTER

Known Uses

- Autonomous Cars Group
- AgileConsultants GmbH
- Retail Corp

C.1.8. Explain Meeting Purpose

Principle Overview	
ID	P-02
Name	EXPLAIN MEETING PURPOSE
Alias	
Summary	In order to align every participant of a meeting, the Agile Coach or Scrum Master shortly summarizes the purpose and goal of an upcoming meeting.
Type	Communication
Binding Nature	Mandatory

Example
During the agile transformation of SoftwareInc many new meeting types have been introduced. Participants are not sure what the purpose, agenda and outcome of those meetings is. Even after some time, there are still people doubting the relevance of agile events, like the Retrospective.

Context
There are many events and meetings within agile frameworks. The purpose of those events is not always clear to everyone.

Problem
C-4: Dealing with doubts in people about changes
C-45: Dealing with black and white mindsets
C-46: Dealing with closed mindedness
C-59: Establishing a common understanding of agile thinking and practices
C-91: Demonstrate Value Add of Agile Methods
C-94: Understanding the Demand for Becoming Agile

Forces
People get demotivated when they do not understand why they need to participate in a meeting. The meeting cannot be efficient.

Consequences
Benefits:

- Every participant understands why the meeting is necessary.
- Meetings are more efficient if no one doubts the importance of the meeting.
- All participants have a common understanding and common expectations.

Liabilities:

- It takes time.
- People, who know the purpose of a meeting, may feel annoyed by this.

See Also

- *M-10: Meeting CheckIn/-Out (Pattern Candidate)*
- *V-08: Golden Circle (Pattern Candidate)*
- *V-11: Meeting Definition (Pattern Candidate)*

Known Uses

- Agile Consultants GmbH
- IT Business Consultancy
- Retail Corp

C.1.9. Good Practice Newsletter

Pattern Overview	
ID	V-02
Name	GOOD PRACTICE NEWSLETTER
Alias	Agile Newsletter, Agile Explained
Summary	Once a week an agile coach publishes everything interesting he saw or read during the last week.
Type	Blog

Example
An Agile Coach from RetailCo reads a lot about new agile practices and successful projects within and outside the organization. He wants to share this information, but does not want to invite everyone to an extra event.

Context
There is no culture of knowledge sharing.

Problem
C-4: Dealing with doubts in people about changes
C-5: Facilitating shared context and knowledge
C-33: Building trust of stakeholders in agile practices
C-39: Establishing a culture of continuous improvement
C-59: Establishing a common understanding of agile thinking and practices
C-91: Demonstrate Value Add of Agile Methods
C-94: Understanding the Demand for Becoming Agile
C-97: Patience during the Agile Transformation
C-109: Learning to improve as an Organization
C-110: Establishing an Agile Mindset

Forces
People tend to not talk about good practices or if their read anything interesting.
There is no culture of information sharing.

Solution
Once a week someone (e.g., an Agile Coach) releases a newsletter via E-Mail containing everything he found interesting this week. The newsletter can contain new methods, visualizations, definitions, information about frameworks, texts about soft skills, or success stories about agile projects. It can be anything agile-related. The intention is purely to share information.

Variants
The newsletter can also be published on a blog or within the intranet.

Consequences
Benefits:

- People do not need to keep them updated themselves.
- People learn that it is fine to share information.
- People learn to actively communicate about things they read or saw if one person starts to do it.

Liabilities:

- People, who are not interested in this information, may feel annoyed by it.

See Also

- P-01: PUBLISH GOOD PRACTICE
- P-03: CELEBRATE EVERY SUCCESS

Data Collection
The agile newsletter is sent out/published once a week, either via mail or on an online collaboration tool.
The information is administrated by an Agile Coach or by a Scrum Master.
All data required for this V-Pattern is illustrated in Figure C.3.

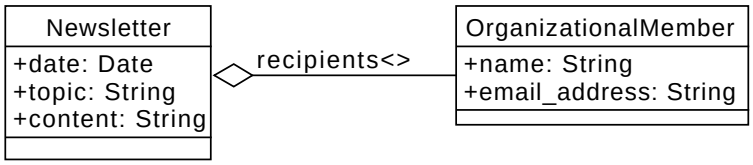


Figure C.3.: Data Model of the Good Practice Newsletter

Known Uses

- Global Insurance Corp
- LuxCars Corp
- Retail Corp

C.1.10. Don't overshoot Coordination Meetings

Anti-Pattern Overview	
ID	A-06
Name	DON'T OVERSHOOT COORDINATION MEETINGS
Alias	
Summary	Too many coordination meetings decrease the efficiency of all teams. Reducing the number by building larger meeting rooms and merging several meetings into one counteract this problem. A predefined weekly meeting schedule, as well as predefined meeting outlines help to set the scope at a minimum meeting number, while still informing everyone.

Example
RetailCo has transformed their processes from traditional to agile. A great number of new, frequent meetings has been established, like the Daily Standup. However, they kept most of the meetings they had before the transformation. Many people complained that there are too many redundant meetings.

Context
A scaling-agile framework has been applied and old meetings have been kept.

Problem
C-49: Dealing with increased efforts by establishing inter-team coordination

Forces
If a meeting takes place at a regular base, people tend to not skip it, even though it is redundant to others.
Everything needs to be communicated to everyone.
There are too few large meetings rooms, so one large meeting is split into several small ones.

General Form
Because of too many coordination meetings, work cannot be done efficiently. Often, several scaling-agile frameworks have been combined, including many coordination meetings. In addition, old coordination meetings that were established before the transformation are still kept.

Consequences
Benefits:

- Everything is communicated.

Liabilities:

- Not every meeting is relevant for every participant.
- Participating irrelevant meetings leads to less time spent to create value for the customer.
- Redundant meetings demotivate people to actively participate in the discussions.

Revised Solution
Set up a regular meeting cycle/schedule and clearly define the participants, time and scope of every meeting. Decide based on the scope, which meetings are really necessary and which can be combined with other meetings. Try to reduce the number as much as possible.
Try to build/find more large meeting rooms (e.g., the canteen) so more people can participate during one meeting.

See Also

- A-7: DON'T USE FRAMEWORKS AS RECIPES
- P-13: EXPLAIN MEETING PURPOSE
- *V-22: Meeting Cycle (Pattern Candidate)*
- *V-23: Meeting Schedule (Pattern Candidate)*

D. Appendix

The following patterns, principles, and anti-patterns have not been identified as real patterns as they were not mentioned in at least three different companies. However, it is possible that other stakeholders use these candidates, which could still make them a pattern. The candidates are documented in the patlet form by Coplien and Harrison [20], which consists of problem-solution pairs. The stars next to a candidate's name indicate the number of mentions by interviewees.

D.1. Documentation of Principle Candidates

ID	Name & Problem	Summary
P-06	*People who are affected by change must be involved in the execution of change**	People must be able to drive their own change. They know best how the organization works, which concepts fit well and which do not. Therefore, Change Management Teams should always include people who are directly affected by changing actions.
	C-81: Enable Change from Process to Product-Orientation	
P-07	*Non-Violent Communication***	In order to build trust and a good working environment, everyone must adhere to the principles of non-violent communication.
	C-24: Creating team spirit and trust among agile teams	
P-08	*Durable Teams**	A team that has worked together for a long time should not be split. Instead, the team should remain a team, even for a new product. The team members know each other well and know how to effectively work together.
	C-24: Creating Team Spirit and Trust among Agile Teams	

ID	Name & Problem	Summary
P-09	*Only invest in tasks affecting the near future**	A team should focus on those tasks that affect the near future, e.g., the next release. It does not make sense to plan months in advance, as change can occur any time. All time invested into planning would be wasted.
	C-72: Considering required competencies when assigning teams to tasks	
P-10	*Modern Agile**	Modern Agile defines four principles that fit a modern, agile world rather than the ones from the manifesto: (1) Make People Awesome, (2) Make Safety a Prerequisite, (3) Experiment and Learn Rapidly, and (4) Deliver Value Continuously [48].
	C-39: Establishing a culture of continuous improvement C-59: Establishing a common understanding of agile thinking and practices	
P-11	*Separate Observation from Solution**	If the teams are too large that a single solution fits right away, you need to observe first and decide on a solution afterwards.
	C-109: Learning to improve as an Organization	
P-12	*Integrate Externals into the Transformation**	Even though external employees are often not seen as part of the organization, they also contribute to value creation and product development. Therefore, keep your externals involved within the agile transformation and let them participate in decisions and events.
	C-83: Dealing with External Developers	

Table D.1.: Principle Patlets

D.2. Documentation of Coordination Pattern Candidates

ID	Name & Problem	Summary
CO-3	*Value Workshop*** C-86: Emotional Consequences of Agile Transformations	This workshop for a team aims at defining common values and emphasizes the individual values of each team member.
CO-04	*Enterprise Retrospective** C-39: Establishing a culture of continuous improvement	An Enterprise Retrospective takes place every two months for one hour with around 100 participants, including the management board, leadership team and as many team members as possible. The goal is to discuss the global impediments and find solutions for them.
CO-05	*Agile Hour** C-91: Demonstrate Value Add of Agile Methods C-110: Establishing an Agile Mindset	The Agile Hour takes place once per sprint for our hour and discusses a predefined topic. The goal is to explain agile concepts, values and practices. The event is organized by the Agile Coach or by the Scrum Master, who also decides the topic. The participation is optional.
CO-06	*Best Practice Webinar** C-59: Establishing a common understanding of agile thinking and practices	In order to guarantee common knowledge and understanding of agile practices and principles in distributed teams, a webinar is conducted, presenting best practices in agile development.
CO-07	*Office Standup** C-49: Dealing with increased efforts by establishing inter-team coordination	In order to inform everyone about current deadlines, updates, impediments, and changes, the whole organization of one site meets every morning for five minutes.
CO-08	*Lean Coffee** C-5: Facilitating Shared Context and Knowledge	To enable knowledge and experience sharing, the organization sets up Lean Coffee sessions, in which different topics can be discussed. Each topic has a discussion time of five minutes and usually relates to agility.

ID	Name & Problem	Summary
CO-09	*Product Owner Leadership Meeting**	The leadership team as well as all Product Owner and relevant team member meet and discuss the status of the product.
	C-102: Motivating Leadership to talk to teams	
CO-10	*Product Owner Alignment**	Twice a week all Product Owner meet and discuss their status, identify inter-dependencies with other teams and set the roadmap. The goal is to set a common vision for all teams working on the same product.
	C-23: Establishing a common scope for different stakeholder groups Establishing a Common Product Vision	
CO-11	*Higher Level Management Update*	During this Coordination Meeting, the Top Management and the Product Owner come together. The Top Owner present their top five topics and define their next five goals.
	C-111: Coaching on Higher Management Level	
CO-12	*Role Clarification**	If conflicts regarding a role's tasks and responsibilities, this coordination meeting can be set up. It takes about two hours and clarifies what different roles do and what they do not. Afterwards, everyone knows what to expect from his colleagues and what is expected by him.
	C-111: Coaching on Higher Management Level C-56: Defining clear Roles and Responsibilities	
CO-13	*Product Owner Roadmap*	This meeting aims at identifying inter-dependencies between teams working on the same product. Each Product Owner presents his five epics for the next two months and the other Product Owner check the epics for possible dependencies with their own epics.
	C-112: Identifying Dependencies between Teams on Cross-Domain Level	
CO-14	*Change Discussion Workshop**	The Agile Coach sets up a series of weekly workshops, where participants discuss one topic. The topic is mostly selected by the Agile Coach, but mostly develops out of prior discussions. The participants are from the management level and should be provided with the chance to simply discuss.
	C-110: Establishing an Agile Mindset C-111: Coaching on Higher Management Level	

Table D.2.: CO-Pattern Patlets

D.3. Documentation of Methodology Pattern Candidates

ID	Name & Problem	Summary
M-04	*Agile Transition Team*** C-81 Enable Change from Process to Product Orientation C-59 Establishing a common understanding of agile thinking and practices	An Agile Transition Team consists of three to five Agile Coaches and all executive managers of an organization. Together they define the goals of the agile transition and facilitate them.
M-05	*Value Stream Optimization** C-81 Enable Change from Process to Product Orientation	This method is a combination of Value Stream Mapping and the Value Proposition Canvas. It identifies pain points in the value stream and tries to dissolve them with possible pain killers from the Value Proposition Canvas.
M-07	*Scrum Master Silo** C-7: Dealing with incorrect practices of agile development	If the Product Owner is the line manager of the Scrum Master, later can not freely talk about problems and incorrect practices. Therefore, all Scrum Masters are placed outside the line organization within a Scrum Master Silo.
M-08	*Working Out Loud** C-19: Dealing with internal Silos	This method consists of a 12 week self coaching within a group of five people. Participants set own goal and need to actively communicate their status to the members of his group as well as actively talk to other people who can help them reach their goal.
M-09	*SCARF Model*** C-87: Patience during the Agile Transformation	Each change triggers emotional consequences within everyone's mind. Change is either a reward or a threat. The SCARF model clarifies an individual's emotions and can be used to explain why someone reacts the way he does.

ID	Name & Problem	Summary
M-10	*Meeting Check-In/CheckOut**	As every meeting participant has they own individual mood that can affect the upcoming meeting, take two minutes in the beginning of each meeting. Within these two minutes, every participant shortly explains his mood, reasons for his mood, and his expectations for the meeting. In the end, take another two minutes for everyone to give feedback on what to improve next time.
	C-39: Establishing a culture of continuous improvement	
M-11	*Agile Coach Training**	Each Agile Coach should have a basic tool set of methods, visualizations and concepts. Therefore, each coach should attend an Agile Coach Training, either at an external consultancy or at an enterprise-internal training.
	C-85: Self-Awareness as an Agile Coach	
M-12	*Team Stabilization**	In order to enable constant measurement of an agile teams performance, set up three KPIs at the beginning of a project and monitor them constantly: (1) a KPI for stability of the scope, (2) a KPI for the stability of the code, and (3) a KPI on how good the predictions made by the team are.
	C-90: Objective Measuring Methods	
M-13	*Agile Fluency Model**	To show teams that the agile transformation takes time and to clarify that they will not be working perfectly from day one, use the Agile Fluency Model. It consists of four steps: (1) forming, (2) delivering, (3) optimizing, and (4) strengthening.
	C-87: Patience during the Agile Transformation	
M-14	*Site Visits**	Distributed teams should get to know each other in person at least once. This increases trust and team spirit among the team members. Therefore, regularly let team members visit other teams on their site.
	C-3: Coordinating geographically distributed agile teams	

ID	Name & Problem	Summary
M-15	*Office Mirroring** C-31: Dealing with geographical distance between agile teams	To facilitate communication between two distributed teams, they set up a video conference for the complete day. For this, each team projects their office, using a web cam, to a wall within the other team's office, and the other way around. Microphones are used to talk to each other. This way, it feels like the teams are co-located as they can see and talk to each other all day.
M-16	*Transform Digital to Analog** C-67: Encouraging development teams to talk about tasks and impediments	Nowadays, all tasks are stored digitally. This blurs the relation of tasks to the real product. Therefore, each morning, each team member writes down their tasks for the upcoming day, making the tasks more tangible.
M-17	*Scrumory** C-59: Establishing a common understanding of agile thinking and practices	A team plays the classic memory game, but with Scrum terms and definitions.
M-18	*Shadowing*** C-5: Facilitating shared context and knowledge C-47: Dealing with higher-level management interferences	When managers want to know how agile practices work in reality, they follow the Scrum Master of a team for one day like a shadow. They participate in every meeting, but are not allowed to speak.
M-19	*Big Picturing** C-109: Learning to Improve as an Organization	As each organization has their own ways of working, these have to be visualized and made explicit. Therefore, create Big Pictures, e.g., 'What do we do?', 'What are our events, time frames and meetings?', and place them on walls within the enterprise for everyone to see.

ID	Name & Problem	Summary
M-20	*Scaling Dance** C-85: Self-Awareness as an Agile Coach	To help Agile Coaches evaluate themselves, place sheets of paper with the numbers from one to twelve on the floor. Each Agile Coach needs to place himself on a certain number and explain his estimation. Afterwards, they think about actions that can bring them to a higher scale.
M-21	*Update Snake** C-39: Establishing a Culture of Continuous Improvement	As not all information can be gathered during one meeting, an Update Snake is installed into the artifact, e.g., a board. An Update Snake is a distinct area within the board, into which participants of the meeting can place anything that came to their mind after the meeting. Two weeks after the meeting has taken place, the participants come together for fifteen minutes and review the items in the Update Snake. This ensures that everything is incorporated into the artifact.

Table D.3.: M-Pattern Patlets

D.4. Documentation of Viewpoint Pattern Candidates

ID	Name & Problem	Summary
V-03	*DeMo Board**	This board visualizes the feedback of a team and enables a culture of continuous improvement. Team members who (anonymously) want to point out good or bad behavior write down an explanation of their (un)fulfilled need and place it on the board. The feedback is visible for everyone and the leadership can react to it.
	C-45: Dealing with Black and White Mindedness C-46: Dealing with Closed Mindsets C-67: Encouraging development teams to talk about tasks and impediments	
V-04	*DISG® Board***	This board is a visualization of the DISG® personality assessment. Each team member writes his initials on a sticky note and places it on the DISG® Board.
	C-44: Dealing with communication gaps with stakeholders C-74: Empowering agile teams to make decisions	
V-05	*Skill Matrix**	The Skill Matrix displays all required skills and the respecting skill level of each team member. It can be used to identify who can carry over someone's work in cross-functional teams and which skills are still required.
	C-89: Establishing equality among cross-functional teams	
V-06	*Problem Solving Matrix**	Even though the Scrum Master is responsible for resolving impediments, the team must not always assign such tasks to him. Instead, each impediment must be classified according to the Problem Solving Matrix. The matrix identified if an impediment can be solved by the team itself, or is forwarded to the Scrum Master.
	C-67: Encouraging development teams to talk about tasks and impediments	

ID	Name & Problem	Summary
V-07	*Release Burn-Up Chart***	The Release Burn-Up Chart visualizes what has already been achieved within the current release, for example how many Story Points or how many Backlog Items.
	C-67: Encouraging development teams to talk about tasks and impediments	
V-08	*Golden Circle***	The Golden Circle is a visualization for explaining the purpose of an agile transformation. First, explain the intention of agile practices, then explain how those practices can be used, and finally describe concrete steps the organization does.
	C-33: Building Trust of Stakeholders in Agile Practices C-94: Understanding the Demand for Becoming Agile	
V-09	*Stakeholder Matrix**	This board manages all stakeholders within four different categories: (1) monitor, (2) keep informed, (3) keep satisfied, and (4) engage fully. Depending on the category, stakeholders require a different level of involvement.
	C-11: Obtaining Management Buy-In	
V-10	*Competence Model**	A Competence Model is an official document set up by experts and the Human Resources Department and defines competences, tasks, and responsibilities of a role. This helps to build a common understanding as well as serves as a foundation for performance evaluation.
	C-16: Dealing with increasing workload of key stakeholders C-56: Defining Clear Roles and Responsibilities	
V-11	*Meeting Definition**	This document defines the goal, participants, time and agenda for all regular meetings.
	C-49: Dealing with increased efforts by establishing inter-team communication	
V-12	*Impact Mapping**	This technique visualizes how each individual member of an organization has an impact onto the agile transformation.
	C-33: Building Trust of Stakeholders in Agile Practices	

ID	Name & Problem	Summary
V-13	*Meeting Schedule** C-49: Dealing with increased efforts by establishing inter-team communication C-78: Synchronizing sprints in the large-scale agile development program	A Meeting Schedule is a time table, which displays all regular meetings (e.g., Daily, Retro, Review) of every team. This way, limited meeting rooms can be planned better and the Lead Product Owner knows how the teams are working.
V-14	*Agile Goal Board** C-73: Dealing with decreased predictability C-106: Agile Program Management	An Agile Goal Board can be set up if the Higher Management does not want to give up on long-tern planning. Therefore, the board consists of three parts: (1) Impediments, (2) Short-Term View, containing Sprint Goals, and (3) Long-Term View, containing future goals, which are not yet addressed in sprints.
V-15	*Learning Backlog** C-39: Establishing a culture of continuous improvement	In order to improve, always transform an impediment into a learning. The Agile Coach or the Scrum Master keeps a Learning Backlog containing former impediments and topics he needs to train his team in.
V-16	*Value Radar** C-110: Establishing an Agile Mindset	The Value Radar is a visualization of a team's important values and the degree of fulfillment of each value.
V-17	*KUDOS Board** C-24: Creating Team Spirit and Trust among Agile Teams	Whenever someone wants to thank someone else or has anything positive to say, he writes his thoughts on a sticky note and sticks it to the KUDOS Board.
V-18	*Transformation Backlog*** C-81: Enable change from process to product-orientation C-109: Learning to improve as an organization	The transformation backlog is set up like a Kanban Board. Every two weeks, the higher management meets in front of the board and reviews the status.

Table D.4.: V-Pattern Patlets

D.5. Documentation of Anti-Pattern Candidates

ID	Name & Problem	Summary
A-2	*Don't Outsource Development***	By outsourcing development tasks, the organization loses know-how and creates large communication and contractual overhead. Instead, internal resources should be build up.
	C-83: Dealing with external Developers	
A-3	*Don't Establish Change Management Teams**	During the Agile Transformation, someone needs to decide on techniques, roles, concepts, etc. Therefore, a Change Management Team consisting of people from the higher management is established. However, these members are not directly affected by the change within the organization. Consequently, only agile practices are adopted, that might not fit the organizational structure. Besides, these changes may not be accepted by the organization. People who are affected by change should also be included within such Change Management Teams.
	C-81 Enable Change from Process to Product Orientation	
A-4	*Don't Include Line Management in Autonomous Teams**	Agile teams are often integrated within a line organization. If a line manager is included within an autonomous team, he often dictates deadlines, wants frequent updates and leads the team in a command & control manner. These are not agile practices and should not be conducted. The team must clear up the line manager's responsibilities and let him participate in their meetings and events for updates.
	C-75: Forming and Aligning Autonomous Teams	
A-5	*Don't Do Shared Code Ownership in Multi-Vendor Scenarios**	In large-scale agile development, multiple vendors may work on the same product. Shared Code Ownership is a concept in which every team member knows the complete code in order to maintain it. However, in multi-vendor scenarios, this concept should not be applied as it violates the principle of Information Hiding and has legal consequences. Use a more fitting architecture, in which not everyone needs to know every detail of the code.
	C-92: Coordination of Multi-Vendor Teams	

ID	Name & Problem	Summary
A-6	*Don't Overshoot Coordination Meetings***	Too many coordination meetings decrease the efficiency of all teams. Reducing the number by building larger meeting rooms and merging several meetings into one counteract this problem. A predefined weekly meeting schedule as well as predefined meeting outlines help to set the scope at a minimum meeting number, while still informing everyone.
	C-49: Dealing with increased efforts by establishing inter-team coordination	
A-7	*Don't Simulate Agility by KPIs**	Financial bonuses for Managers are calculated based on their performance. In agile teams, performance cannot be measured against KPIs. However, Managers still set up a set of KPIs to be used for their bonus. Consequently, they simulate agility and good performance. Therefore, revise the bonus system within the company and tailor it to the agile environment.
	C-47: Dealing with higher-level management interferences	
A-8	*Don't Establish Rewarding Models**	Rewarding models aim at motivating individuals to increase their performance. However, by such systems, team members stop working for a higher purpose, i.e., the product, customer satisfaction, and value creation. Therefore, stop using rewarding models and create a social environment, in which people want to work productively.
	C-55: Creating a teamwork centric rewarding model	
A-9	*Don't Include only a few people per meeting**	Managers are afraid to steal other people's time. Therefore, they only invite one or two people (e.g., only the Product Owner and the Scrum Master) to a meeting. However, the team members also need to be included.
	C-109: Learning to improve as an Organization C-49: Dealing with increased efforts by establishing inter-team communication	

ID	Name & Problem	Summary
A-10	*Don't Include Pseudo Product Owner**	During the agile transformation, managers need to decide on a role. Because of different reputations of each role, most managers decide to become Product Owner, even though they do not have the skills and mindset. Do not let everyone become a Product Owner. Create incentives for people to take a different role.
	C-56 Defining clear roles and responsibilities	

Table D.5.: Anti-Pattern Patlets

Bibliography

[1] Disciplined Agile. Disciplined Agile - The Foundation for Business Agility. http://disciplinedagiledelivery.com [Online; accessed 09-May-2019].

[2] Christopher Alexander. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, 1977.

[3] Christopher Alexander. *The Timeless Way of Building*. Oxford University Press, New York, 1979.

[4] 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.

[5] Julian M. Bass. Scrum Master Activities: Process Tailoring in Large Enterprise Projects. In *2014 IEEE 9th International Conference on Global Software Engineering (ICGSE)*, pages 6–15, Washington, USA, 2014.

[6] Kent Beck, Mike Beedle, Arie Van Bennekum, and Alistar Cockburn. *Manifesto for Agile Software Development*. 2001.

[7] Mike Beedle, James O. Coplien, Jeff Sutherland, Jens C. Østergaard, Ademar Aguiar, and Ken Schwaber. Essential Scrum Patterns. https://www.hillside.net/plop/2010/papers/beedle.pdf [Online; accessed 14-May-2019].

[8] Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, and Jeff Sutherland. SCRUM: An Extension Pattern Language for Hyperproductive Software Development. 1999. http://jeffsutherland.org/scrum/scrum_plop.pdf [Online; accessed 14-May-2019].

[9] Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, and Jeff Sutherland. SCRUM: A Pattern Language for Hyperproductive Software Development. In Neil Harrison, Brian Foote, and Hans Rohnert, editors, *Pattern Languages of Program Design 4*, pages 637–651. Addison Wesley Longman, 2000.

[10] Ilya Bibik. *How to Kill the Scrum Monster*. Quick Start to Agile Scrum Methodology and the Scrum Master Role. Apress, Montreal, Canada, 2018.

[11] Susan A. Brown and Yulia W. Sullivan. Guidelines for Conducting Mixed-methods Research: An Extension and Illustration. *Journal of the Association for Information Systems*, 27(7):435–494, 2016.

[12] William H. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick III, and Thomas J. Mowbray. *Anti Patterns*. Refactoring Software, Architectures, and Projects in Crisis. John Wiley & Sons, Canada, 2002.

[13] Sabine Buckl, Florian Matthes, Alexander W. Schneider, and Christian M. Schweda. Pattern-based design research - an iterative research method balancing rigor and relevance. In *DESRIST*, 2013.

[14] Frank Buschmann, Regine Meunier, Peter Sommerlad, Michael Stal, and Hans Rohnert. *Pattern-oriented Software Architecture*. A System of Patterns. John Wiley & Sons, 1996.

[15] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. *Pattern-oriented Software Architecture: A Pattern Language for Distributed Computing*, volume 4. John Wiley & Sons, 2007.

[16] The LeSS Company B.V. Graphics from the less.works website. `https://less.works/resources/graphics/site-graphics.html` [Online; accessed 09-May-2019].

[17] James O. Coplien. A generative development-process pattern language. In James O. Coplien and Douglas C. Schmidt, editors, *Pattern Languages of Program Design*, pages 183–237. ACM Press/Addison-Wesley Publishing Co., New York, USA, 1995.

[18] James O. Coplien. *Software Patterns*. The Hillside Group, 1996.

[19] James O Coplien and Gertrud Bjørnvig. *Lean Architecture for Agile Software Development*. John Wiley and Sons, Padstow, Cornwall, Great Britain, 2010.

[20] James O. Coplien and Neil B. Harrison. *Organizational Patterns of Agile Software Development*, volume 1. Prentice Hall Inc., Upper Saddle River, USA, 2004.

[21] Daniela S. Cruzes and Tore Dyba. Recommended steps for thematic synthesis in software engineering. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 275–284. IEEE, 2011.

[22] Ward Cunningham. Portland pattern repository, 2009. `http://c2.com/ppr/` [Online; accessed 14-May-2019].

[23] Rachel Davies and Liz Sedley. *Agile Coaching*, volume 1. O'Reilly UK Ltd., USA, 2009.

[24] English Oxford Living Dictionaries. Pilot, 2019. `https://en.oxforddictionaries.com/definition/pilot` [Online; accessed 06-May-2019].

[25] Philipp Diebold, Jan-Peter Ostberg, Stefan Wagner, and Ulrich Zendler. What Do Practitioners Vary in Using Scrum? In *Agile Processes in Software Engineering and Extreme Programming*, pages 40–51. Springer International Publishing, Cham, 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 and Nils Brede Moe. Towards Principles of Large-Scale Agile Development. In *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*, pages 1–8. Springer International Publishing, Cham, 2014.

[28] Torgeir Dingsøyr, Sridhar Nerur, Venugopal Balijepally, and Nils Brede Moe. A Decade of Agile Methodologies - Towards Explaining Agile Software Development. *Journal of Systems and Software*, 85(6):1213–1221, 2012.

[29] Torgeir Dingsøyr, Tor Erlend Fægri, and Juha Itkonen. What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development. In *Software Architecture*, pages 273–276. Springer International Publishing, Cham, 2014.

[30] Veli-Pekka Eloranta, Marko Leppänen, and Kai Koskimies. Using domain model for structuring pattern language. page 15, 2015.

[31] Amr Elssamadisy. *Agile Adoption Patterns: A Roadmap to Organizational Success*. Addison-Wesley Professional, 2008.

[32] Alexander M. Ernst. *A Pattern-based Approach to Enterprise Architecture Management*. PhD thesis, Technische Universität München, Munich, 2010.

[33] Steven Fraser, Erik Lundh, Rachel Davies, Jutta Eckstein, Diana Larsen, and Kati Vilkki. Perspectives on Agile Coaching. In Pekka Abrahamsson, Michele Marchesi, and Frank Maurer, editors, *Proceedings of the 10th International Conference on Agile Processes in Software Engineering and Extreme Programming*, pages 271–276, 2009.

[34] Boris Gloger. *Scrum*. Produkte zuverlässig und schnell entwickeln. Carl Hanser Verlag GmbH Co KG, 2016.

[35] Boris Gloger and Jürgen Margetich. *Das Scrum-Prinzip - Agile Organisationen aufbauen und gestalten*. Schäffer-Poeschel Verlag, Stuttgart, 2014.

[36] Daniel R. Greening. Enterprise Scrum: Scaling Scrum to the Executive Level. In *Proceedings of the 43rd Hawaii International Conference on System Sciences*, 2010.

[37] Egon G. Guba. Criteria for Assessing the Trustworthiness of Naturalistic Inquiries. *Educational Communication and Technology*, 29(2):75–91, 1981.

[38] Neil B. Harrison. Organizational Patterns for Teams. In John Vlissides, James O. Coplien, and Norman L. Kerth, editors, *Pattern Languages of Program Design*, pages 345–352. Reading, MA, 1996.

[39] Neil B. Harrison. Advanced Pattern Writing - Patterns for Experiences Pattern Authors. In Dragos Manolescu, Markus Voelter, and James Noble, editors, *Pattern Languages of Program Design 5*. Addison-Wesley, 2006.

[40] Jeanette Heidenberg, Mari Matinlassi, Minna Pikkarainen, Piia Hirkman, and Jari Partanen. Systematic Piloting of Agile Methods in the Large: Two Cases in Embedded Systems Development. In *Software Architecture*, pages 47–61. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[41] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1):75–105, 2004.

[42] Scaled Agile Inc. Safe® for lean enterprises, n.d.. `https://www.scaledagileframework.com/` [Online; accessed 09-May-2019].

[43] Scaled Agile Inc. Value streams, n.d.. `https://www.scaledagileframework.com/value-streams/` [Online; accessed 09-May-2019].

[44] Scrum Inc. Scrum of scrums, n.d.. `https://www.scruminc.com/scrum-of-scrums/` [Online; accessed 10-May-2019].

[45] International Standardization Organization. ISO/IEC/IEEE 42010:2011(E), Systems and software engineering — Architecture description, 2011.

[46] Hanna Kallio, Anna-Maija Pietilä, Martin Johnson, and Mari Kangasniemi. Systematic methodological review: developing a framework for a qualitative semi-structured interview guide. *Journal of Advanced Nursing*, 72(12):2954–2965, 2016.

[47] Allan Kelly. *Changing Software Development: Learning to Become Agile*. John Wiley & Sons, 2008.

[48] Joshua Kerievsky. An Introduction to Modern Agile. 2016. `https://www.infoq.com/articles/modern-agile-intro` [Online; accessed 14-May-2019].

[49] Pouya A. Khosroshahi, Matheus Hauder, Alexander W. Schneider, and Florian Matthes. Enterprise Architecture Management Pattern Catalog V2. Technische Universität München, 2015.

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

[51] Henrik Kniberg and Anders Ivarsson. Scaling Agile @ Spotify. 2012. `https://creativeheldstab.com/wp-content/uploads/2014/09/scaling-agile-spotify-11.pdf` [Online; accessed 14-May-2019].

[52] Maarit Laanti. Characteristics and Principles of Scaled Agile. *XP Workshops*, 199 (Chapter 2):9–20, 2014.

[53] Craig Larman and Bas Vodde. Scaling Agile Development. *CrossTalk*, pages 8–12, 2013.

[54] Craig Larman and Bas Vodde. *Large-Scale Scrum*. More with LeSS. Addison-Wesley, 2017.

[55] Erik Lundh. Elements of an Art - Agile Coaching. In *Proceedings of the 10th International Conference on Agile Processes in Software Engineering and Extreme Programming*, pages 238–239, 2009.

[56] Aniket Mahanti. Challenges in Enterprise Adoption of Agile Methods – A Survey. *Journal of Computing and Information Technology*, 14(3):197–206, 2006.

[57] Christoph Mathis. *SAFe Das Scaled Agile Framework*. Lean und Agile in großen Unternehmen skalieren. dpunkt.verlag, Heidelberg, 1 edition, 2016.

[58] Peter Measey. *Agile Foundations: Principles, Practices and Frameworks*. BCS, The Chartered Institute for IT, London, 2019.

[59] Gerard Meszaros and Jim Doble. A pattern language for pattern writing. Pattern Languages of Program Design 3, pages 529–574. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1997.

[60] Ian Mitchell. *Agile Development in Practice*. TamaRe House, 2016.

[61] Mario E. Moreira. *The Agile Enterprise*. Building and Running Agile Organizations. Apress, Berkeley, USA, 2017.

[62] Ashish Mundra, Sanjay Misra, and Chitra A Dhawale. Practical Scrum-Scrum Team: Way to Produce Successful and Quality Software. In *2013 13th International Conference on Computational Science and Its Applications (ICCSA)*, pages 119–123. IEEE, 2015.

[63] Maria Paasivaara and Casper Lassenius. Communities of practice in a large distributed agile software development organization - Case Ericsson. *Information and Software Technology*, 56(12):1556–1577, 2014.

[64] Ken Pfeffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–78, 2007.

[65] Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison Wesley, 2003.

[66] Ken Power. A Model for Understanding When Scaling Agile Is Appropriate in Large Organizations. In T Dinsor and N Bremede, editors, *Lecture Notes in Business Information Processing XP2014 Workshops. vol. LNBIP 199*, pages 83–92. Springer, 2014.

[67] Viviane Santos, Alfredo Goldman, and Heitor Roriz Filho. The Influence of Practices Adopted by Agile Coaching and Training to Foster Interaction and Knowledge Sharing in Organizational Practices. *2013 46th Hawaii International Conference on System Sciences*, pages 4852–4861, 2012.

[68] Roman Sauter, Werner Sauter, and Roland Wolfig. *Agile Werte- und Kompetenzentwicklung*. Wege in eine neue Arbeitswelt. Springer-Verlag, Berlin, Heidelberg, 2018.

[69] Ken Schwaber. Online Nexus Guide, 2018. `https://www.scrum.org/resources/online-nexus-guide` [Online; accessed 09-May-2019].

[70] Ken Schwaber and Jeff Sutherland. The Scrum Guide, 2017. [`https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf` Online; accessed 09-May-2019].

[71] ScrumPLoP. Scrumplop, 2019. `https://sites.google.com/a/scrumplop.org/published-patterns/home` [Online; accessed 25-April-2019].

[72] Andrew K. Shenton. Strategies for ensuring trustworthiness in qualitative research projects. *Education for Information*, 22:63–75, 2004.

[73] Ahmed Sidky and James Arthur. A Disciplined Approach to Adopting Agile Practices: The Agile Adoption Framework, 2007. `https://arxiv.org/pdf/0704.1294.pdf` [Online; accessed 09-May-2019].

[74] Durward K. Sobek and Cindy Leduc Jimmerson. A 3 Reports : Tool for Organizational Transformation. Houston, Texas, 2004.

[75] Paul Taylor. Capable, Productive, and Satisfied: Some Organizational Patterns for Protecting Productive People. In Neil Harrison, Brian Foote, and Hans Rohnert, editors, *Pattern Languages of Program Design 4*. Addison Wesley Longman, 2000.

[76] Kevin Thompson. Recipes for Agile Governance in the Enterprise, 2016. `ttps://www.cprime.com/wp-content/uploads/woocommerce_uploads/2013/07/RAGE-Final-cPrime1.pdf` [Online; accessed 10-May-2019].

[77] Oktay Turetken, Igor Stojanov, and Jos J. M. Trienekens. Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework. *Journal of Software: Evolution and Process*, 29(6):1796–18, 2016.

[78] Ömer Uludağ, Martin Kleehaus, Christoph Caprano, and Florian Matthes. Identifying and Structuring Challenges in Large-Scale Agile Development Based on a Structured Literature Review. In *IEEE nd International Enterprise Distributed Object Computing Conference*, Stockholm, 2018. EDOC.

[79] Antti Välimäki. *Pattern Language for Project Management in Global Software Development*. PhD thesis, Tampere University of Technology. Publication; Vol. 970, Tampere University of Technology, 2011.

[80] VersionOne. 13th annual State of Agile Report, 2019.

[81] Xiaofeng Wang, Kieran Conboy, and Oisin Cawley. "Leagile" Software Development: An Experience Report Analysis of the Application of Lean Approaches in Agile Software Development. *The Journal of Systems Software*, pages 1–28, 2010.

[82] Tim Wellhausen and Andreas Fiesser. How to write a pattern? In *EuroPLoP*, pages 1–9, New York, USA, 2011. ACM Press.

[83] Etienne Wenger, Richard McDermott, and William M. Snyder. *Cultivating Communities of Practice*. Harvard Business School Press, Boston, USA, 2002.

[84] Adam Wiggins. The twelve-factor app, 2017. `https://12factor.net/` [Online; accessed 11-May-2019].

[85] Laurie Williams and Alistair Cockburn. Agile Software Development: It's about Feedback and Change. *IEEE Computer*, 36:39–43, 2003.

[86] Ralf Wirdemann and Johannes Mainusch. *Die Grundlagen von Scrum*. Carl Hanser Verlag GmbH & Co. KG, 2017.