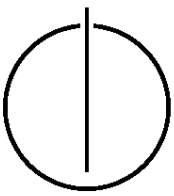
DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Observing the Learning Process of a Large-Scale Agile Development Program - A Case Study from the Technology Sector

Niels Holz



DEPARTMENT OF INFORMATICS

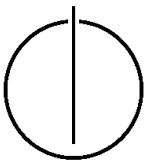
TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Observing the Learning Process of a Large-Scale Agile Development Program - A Case Study from the Technology Sector

Beobachtung des Lernprozesses eines skalierten agilen Programms - Eine Fallstudie im Technologiesektor

Author:	Niels Holz
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	M. Sc. Ömer Uludağ
Submission Date:	March 15, 2020



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

Munich, March 16, 2020 Niels Holz

Acknowledgments

Thank you to everybody supporting my research at Technology LLC., without your support and eagerness to help me, this thesis would not have been possible. A special thanks to all program members letting me sit in on meetings and making notes. Of course, once more, thank you to all my interview partners and everybody involved in the implementation of the Community of Practice.
Thank you Ömer for having patience during the longer than expected search for a case study partner and then of course for your continued support throughout my thesis.
Thank you Prof. Dr. Matthes for supervising this thesis.
Finally I want to thank my family and friends for helping me wherever they could! Special thanks to my sister and my brother-in-law for proofreading my thesis.

Abstract

Agile development has been on the rise in the last couple of years. A Concept created for small, co-located teams to boost their development process. Seeing the success, large organizations are more and more interested to apply agile practices themselves. Projects using agile practices manage to deliver new product increments quicker and with a higher customer satisfaction. Some new challenges arise through scaling agile, with their cause in the structure of larger organizations, others are based around the issue of working in geographical distributed teams, working with legacy systems and having a broad set of customers. While there is research on challenges of large-scale agile development and documenting good practices available. There is only little publication on implementing observed practices and observing lessons learned at other organizations.
The documentation of the as-is situation done in this thesis is based on available research on different concepts of large-scale agile development and a Large-Scale Agile Development Pattern Language for identifying recurring concerns and documenting good and bad practices. With a detailed picture of the as-is situation an analysis of recurring concerns and problem areas of the case study partner is possible. Using the already documented good practices from other research, this thesis then provides the case study partner with possible fitting patterns to address these concerns. In a final step the introduction of one presented pattern is observed.
This thesis uses a mixture of exploratory and descriptive case study methodologies. Three interviews with case study partners were conducted to document the as-is situation of their large-scale agile development and eleven semi-structured interviews provided the documentation of recurring concerns and good practices for the stakeholders groups Product Owner, Scrum Master and Development Team. These interviews led to the discovery of 50 pattern candidates, one coordination pattern and one anti-pattern, using the *Rule of Three*. Further, this thesis introduces four coordination patterns, one methodology patterns and one principle to address the problem areas of the Large-Scale Agile Development Program. Out of the introduced six patterns, this thesis then goes on to implement and evaluate one coordination pattern at the case study partner.
This thesis is part of a major research project to create a pattern catalog for Large-Scale Agile Development. As part of this research project, this thesis tries to verify, by example, already discovered patterns of the pattern catalog and document the deviations of the application in practice.

Contents

List of Figures

List of Tables

List of Abbreviations

CoP	Community of Practice
CO-Pattern	Coordination-Pattern
DoD	Definition of Done
DoR	Definition of Ready
DEV	Development Team
DDD	Domain-Driven-Design
EAM	Enterprise Architecture Management
LSAD	Large-Scale Agile Development
LSADPL	Large-Scale Agile Development Pattern Language
LSADP	Large-Scale Agile Development Program
LeSS	Large-Scale Scrum
M-Pattern	Methodology-Pattern
MVP	Minimum Viable Product
PBR	Product Backlog Refinement
PO	Product Owner
SM	Scrum Master
SoS	Scrum of Scrums
sebis chair	Chair of Software Engineering for Business Information Systems
V-Pattern	Viewpoint-Pattern

Outline of the Thesis

CHAPTER 1: INTRODUCTION

The first chapter presents the motivation of the thesis. It explains the Research Questions and introduces the Research Approach.

CHAPTER 2: FOUNDATIONS

The Foundations define important terms and present the context in which this thesis is placed. The chapter summarizes existing research by the sebis chair that this thesis builds on. Also, the theoretical foundations of agile methods, pattern-based solution approaches, and coordination theory are presented.

CHAPTER 3: RELATED WORK

The third chapter compares several pattern languages in the agile and large-scale context. Further, it discusses related work on coordination in large-scale agile software development.

CHAPTER 4: CASE STUDY

This chapter presents the case study that is conducted in the thesis. First the case study partner is introduced and the method for data collection is presented. Followed, by the findings on the agile transformation, the recurring concerns identified and best and bad practices documented. Finally this chapter documents th implementation process of the instantiated pattern at the cases study partner and the findings.

CHAPTER 5: DISCUSSION

The Discussion outlines the key findings of the thesis and discusses the Limitations of this thesis.

CHAPTER 6: CONCLUSION

Finally this chapter summarizes the work and mentions potential future work.

1. Introduction

1.1. Motivation

While the Agile Manifesto is a collection of principles and values to adhere to when working in agile development, the research following its publication led to frameworks and practices describing how to work in agile development. Agile practices and agile teams create better products with higher quality and customer satisfaction, while being able to react quickly to changing requirements[23].
Software development using the waterfall or v-model focuses on detailed planning for each process step and following the respective process proposed by the model.
V-model and waterfall should be chosen over agile practices, when working in a large-scale development program with clear requirements. Agile practices are more useful for short time horizons and co-located small teams [5].
The statement " agile practitioners and other researchers concluded that the agile value set and practices best suit co-located teams of about 50 people or fewer who have easy access to user and business experts and are developing projects that are not life-critical"[65], has been challenged by the emergence of Large-Scale Agile Development (LSAD).
These larger organizations hope to emulate the success stories observed in teams working with agile practices. In recent years, studies have shown the circumstances of a project or program do not limit the use of agile practices. Version One's 13th annual state of agile report[27] shows that 72% of 1319 surveyed organizations, of which 46% have more than 5000 employees, practice some sort of Scrum. Through adapting agile, larger organizations want to reduce project costs and risks, increase the product quality and be able to adapt to changing requirements[27]. One major trend is to adopting some sort of scaled agile practice. Larger organizations are motivated by the successes observed when applying agile practices. As a result they scale agile practices to fit their construct[22].
When applying agile practices at scale, the larger organizations' structures introduce impediments, which have to be dealt with in LSAD. With LSAD, multiple teams work on the same product. These teams can be geographically distributed, therefore, agile practices are also used to address process problems on the team-level [29, 43]. Dikert et al.[18] identified 29 success factors and a total of 35 challenges for agile transformation. With the challenges for LSAD having a multitude of origins, one example for an identified challenge is the acceptance and application of the agile mindset [18]. In a literature review, the Chair of Software Engineering for Business Information Systems(sebis chair) identified multiple recurring concerns from different stakeholder groups, which were thereupon categorized[57]. On the basis of these concerns, further research has identified a fitting Large-Scale Agile

Development Pattern Language (LSADPL) and documented several patterns addressing the aforementioned concerns[**uludaug2019documenting**]. Both of which will be further discussed in Section2.3 and overall in Chapter2. Of the identified concerns by Uludağ et al.[57], eight are connected to the Development Team, fifteen connected to Product Owners and eleven directly connected to Scrum Masters.
Therefore, this thesis focuses first on identifying other concerns of these three stakeholder groups. Secondly, the thesis will identify patterns addressing these concerns and in a third step it will introduce already observed patterns from the pattern catalog[61] at the case study partner, implementing one coordination pattern.

1.2. Research Objectives

This thesis consists of five research questions aimed at analyzing the as-is situation of the applied and adapted Large-Scale Scrum(LeSS) and Spotify model at the case study partner.
RQ1. How has LeSS been adopted and applied at the case study partner?
The purpose of research question RQ1 is to introduce the case study partner's Large-Scale Agile Development Program (LSADP). The goal is a graphical depiction of the LSAD process together with identifying and defining involved roles, events and artifacts. Additionally, identifying the agile principles and values pursued by case study partner. **RQ2. What are recurring concerns of stakeholders at the product organization of the case study partner?**
Based on the roles identified, matching stakeholder groups will then be examined regarding their recurring concerns. The goal is to validate which recurring concerns from Uludağ et al.[57] occur in practice as well as adding potentially new recurring concerns.
RQ3. What are good practices for addressing recurring concerns of stakeholders of the product organization of the case study partner?
Combining the identified recurring concerns to good practices at the case study partner. These identified good practices are then documented using the LSADPL from Uludağ et al.[**uludaug2019documenting**].
RQ4. Which bad practices should be avoided in the product organization of the case study partner?
In addition, bad practices or so called Anti-Patterns are also documented using the LSADPL from Uludağ et al.[55].
RQ5. What are the lessons learned of implementing already observed best practices in the product organization of the case study partner?
Finally on the basis from the findings of RQ2, RQ3 and RQ4, this thesis will introduce best practices documented in the pattern catalog[61] and observe the lessons learned from introducing these patterns at the case study partner.

1.3. Approach

In this thesis, two research approaches have been combined. A descriptive case study to document the as-is situation of the case study partners' LSADP and an explorative case study approach to test the implementation of already applied patterns at the case study partner. For the descriptive case study, a combination of structured interviews and meeting observations were used. The data described the as-is situation of the LSADP as well as the agile transformation process of the case study partner. The explorative case study has been conducted according to the process of Pattern-based Design Research described by Buckl et al.[10]. As highlighted in Figure1.1 this thesis is documenting patterns from practice and applies patterns from the theory in the practice. The Pattern-based Design Research this thesis follows, for identifying recurring concerns (see Section4.3) as well as patterns (see Section4.4) addressing these concerns can be seen in Figure1.1. The Pattern-based Design Research by Buckl et al. [10] consists of four stages:

1. **Observe & Conceptualize** - following a pattern structure to document good practices.
2. **Pattern-based Theory Building and Nexus Instantiation** - describing the evolving process into design theories.
3. **Solution Design & Application** - solution design to configure the solution of a pattern to organizations needs.
4. **Evaluation & Learning** - observe deviations from configured design and instantiated solution and learn from deviations.

This thesis is applying the LSADPL, designed Uludağ et al.[55], for the identification of recurring concerns, for the three stakeholder groups, Development Team, Product Owner and Scrum Master, document good and bad practices and use patterns of the pattern catalog documented with the LSADPL for implementation at the case study partner.
Section2.3 will discuss in detail why and how the LSADPL used in this thesis was chosen and what it looks like. This method was conducted and supported by semi-structured interviews, aimed at identifying good practices at the case study partner. Meeting and work routine observations supplement this method further. Interview partners were presented with the findings of the completed semi-structured interviews to introduce patterns observed at other organizations. Introducing six patterns from the pattern catalog at the case study partners' LSADP. In Cooperation with the case study partner, these patterns were analyzed to fit their LSADP. The response to the implementation of the chosen pattern was captured in Section4.5. A final questionnaire summarizes the lessons learned from introducing patterns, implementing one, of the pattern catalog at the case study partner. The final questionnaire delivers a quantifiable notion towards the applicability of the implemented pattern from the pattern catalog5.
Overall, this thesis aims to apply the following steps of the Pattern-Based Design Research,

Observe and Conceptualize, **Solution Design** and **Evaluation and Learning**.
These steps are represented in the thesis as follows:
First, applied patterns were observed at the case study partners' LSADP in Section4.4.
Second, appropriate solutions were identified and designed for the LSADP in Section4.5.
Finally, the lessons learned from the application of the solutions presented were evaluated in Section4.5.7. Focusing on the deviations between configured design and instantiated solution.

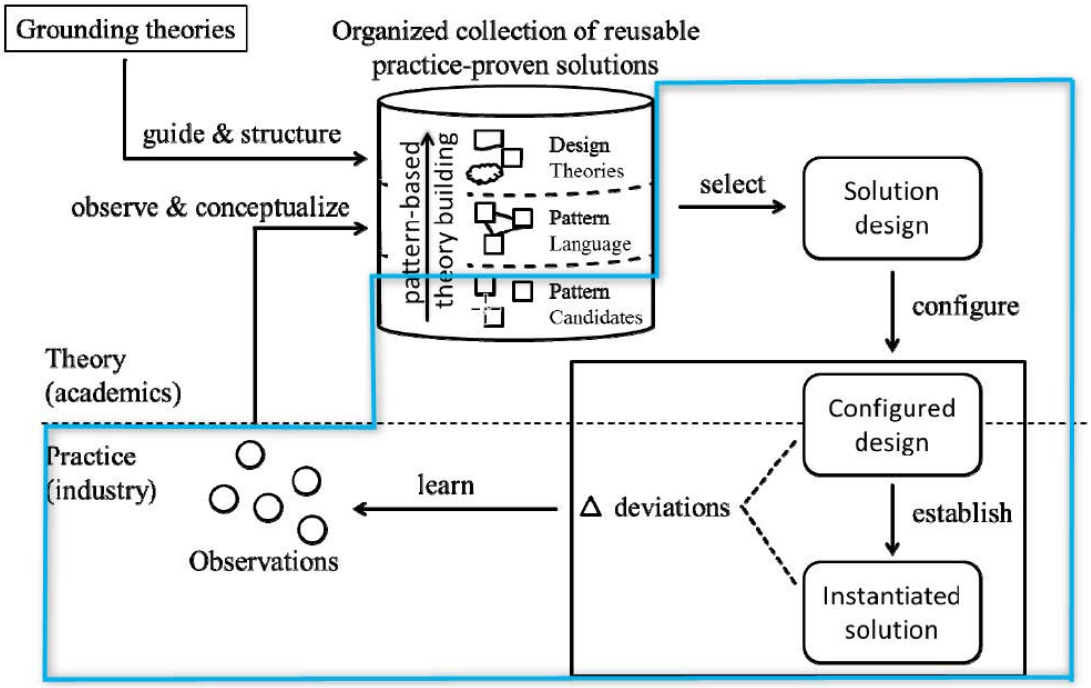


Figure 1.1.: Pattern-Based Design Research[10]

2. Foundations

In this chapter we will discuss different foundations for this thesis. The goal of this chapter is to create a frame for the findings of this thesis.
Therefore, we are revisiting the theoretical aspects of agile software development and the underlying agile principles (see Section2.1) first.
The topic of LSAD offers several different frameworks to tackle many concerns of the real work life. In Section2.2 we will discuss some of the frameworks applied at the case study partner. Additionally we take a look at LSADPL and other Software Development Pattern Languages, in Section2.3. The for this thesis relevant stakeholder groups of LSADP are introduced in the Section2.3.4.

2.1. Agile Software Development

This section introduces the concept of Agile Software Development. Beginning with the Definitions and Values(see Section2.1.1) associated with Agile Software Development. Additionally, Lean Development (see Section2.1.2) principles are introduced and differentiated from Agile Software Development. Finally, this section introduces the method of Scrum (see Section2.1.3) as most large-scale agile methods have their foundation in scaling Scrum.

2.1.1. Definitions and Values

Agile is a frequently used term in recent years, however, it is not always used with the same definition in mind. In agile software development, the definition for the term 'Agile' is a set of principles and values, outlined in the 'Agile Manifesto'[7]. Opposed to traditional software development practices, where the focus lies on processes and methods to achieve a long-term planned development goal[9], in Agile Development a set of principles and values are the core and a set of agile practices are used to support the Agile Development process. The influence of these principles and values cannot be underrated, as the agile practice would not exist and function without them. Agile Development aims at satisfying the customers with short cycle times, each delivering the highest possible quality shippable product increments.[46]. Through these agile practices, the focus shifted from linear development iteration-based development. In agile practices, requirements are constantly revisited, so by the time they go into production, they are specified to most accurately represent a customer need[12]. This iterative adaption and possibility to adapt

to change is one of the key benefits agile practices offer over traditional practices. With traditional projects, requirements are set in stone, which has led to projects realizing in their final stages that the initially created concept or solution does not represent the actual customer need[46]. As depicted in Figure2.1 projects using the Waterfall model have

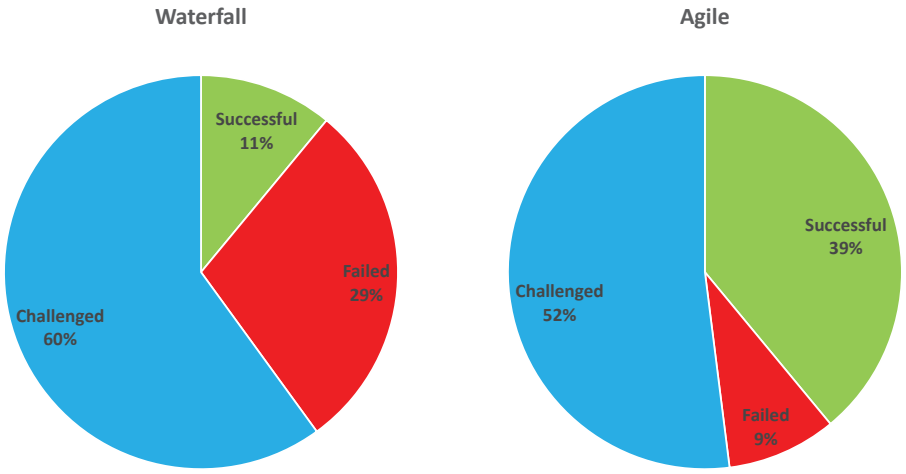


Figure 2.1.: CHAOS Report 2015 - Success rate all project sizes[54]

a three times higher rate of failed projects, than projects using agile practices[54]. The CHAOS report summarizes responses from over 10.000 software projects in the span of five years[54]. Reports like the CHAOS Report of 2015 highlight the achievable success agile practices provide. Agile practices focus on implementing highest priority features first, as these are usually the most challenging and need to be revised more often. These high priority features are implemented in the next development cycle. This incremental approach allows for immediate feedback and resolving the most important features early on in a project, generating higher customer satisfaction[46]. Without using a fixed schedule, agile practices are able to adapt to changing requirements or circumstances with more ease. These principles and values have been published in the 'Agile Manifesto'[7] by 17

authors, who committed themselves to change their approach to software development. In total, the 17 authors proposed four values and twelve principles[7] to adhere to when working in agile software development. The values of the 'Agile Manifesto'[7] are as follows:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

These statements focus on shifting the importance in software development from the right of each statement to the left[7]. These four values aim at creating higher motivation for team members, lightweight documentation, customer involvement and adaptable plans [37]. In accordance with these shifted values, the authors have generated a mindset, which can create benefits measurable in the amount of delivered features as well as higher software quality and higher motivation among team members. Based on these four core values of agile software development, the authors of the 'Agile Manifesto'[7]further define twelve principles, which need to be adhered to.

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.

Principles and Values of the 'Agile Manifesto' focus on the mindset of all team members as well as a customer-centric approach to software development. While the 'Agile Manifesto' delivered the values and principles,frameworks building on the 'Agile Manifesto' interpreted the usability of agile practices for small teams consisting of four to ten people[36], preferably working co-located[37]. Due to these *limitations* the core values of direct communication and focus on people and customers remain the center of agile practices. It is important to avoid pseudo agility, in which employees have not adapted agile values of the "Agile Manifesto"[57]. Employees who follow and believe in the agile mindset lead to more successful projects.

2.1.2. Lean & Agile Development

The concept of Lean is often used in connection with Agile Development. The term Lean is associated with the work of Liker in 'The Toyota Way'[30], which introduced the lean manufacturing practices applied at Toyota. When lean thinking, which similar to the "Agile Manifesto" represents a shift in organizational priorities from the traditional practices, is introduced organizations redefine their approach of software development. The essence of lean thinking is focused around 'building people over building products'[30]. In Lean a lot of concepts of people management and overall project management are challenged. The focus of Lean is to minimize waste originating from three sources(see Larman[32]):

1. Variability - verification at iteration length, feature sizes and team composition.
2. Overburden - overtime for arbitrary deadlines, bottlenecks and dependency on these bottlenecks.
3. Non-Value-Adding actions - hand-offs, task switching, partially done work.

In Combination with avoiding these sources of waste, Lean Development constantly challenges the status quo of the current development process. When working with a lean mindset, one should never assume the current process is set in stone. Rather the individuals are encouraged to challenge the process, to make it even more fitting for the team. Principles of lean are similar to agile principles, the main principles of lean consist of the elimination of waste, a focus on fast delivery, making decisions as late as possible and for managers to go visit employees and be closer to work done on a day-to-day basis[42]. Larman and Vodde[30] summarize the Lean Development concept in the Lean Thinking House, see Figure2.2 The Lean Thinking House(see Figure2.2) shows the pillars and core principles of Lean Development. Continuous Improvement, Respect for People, Management Support and Customer-Centric Development are the pillars of Lean Development(compare Figure2.2). These pillars are similar to the agile values. In addition, the

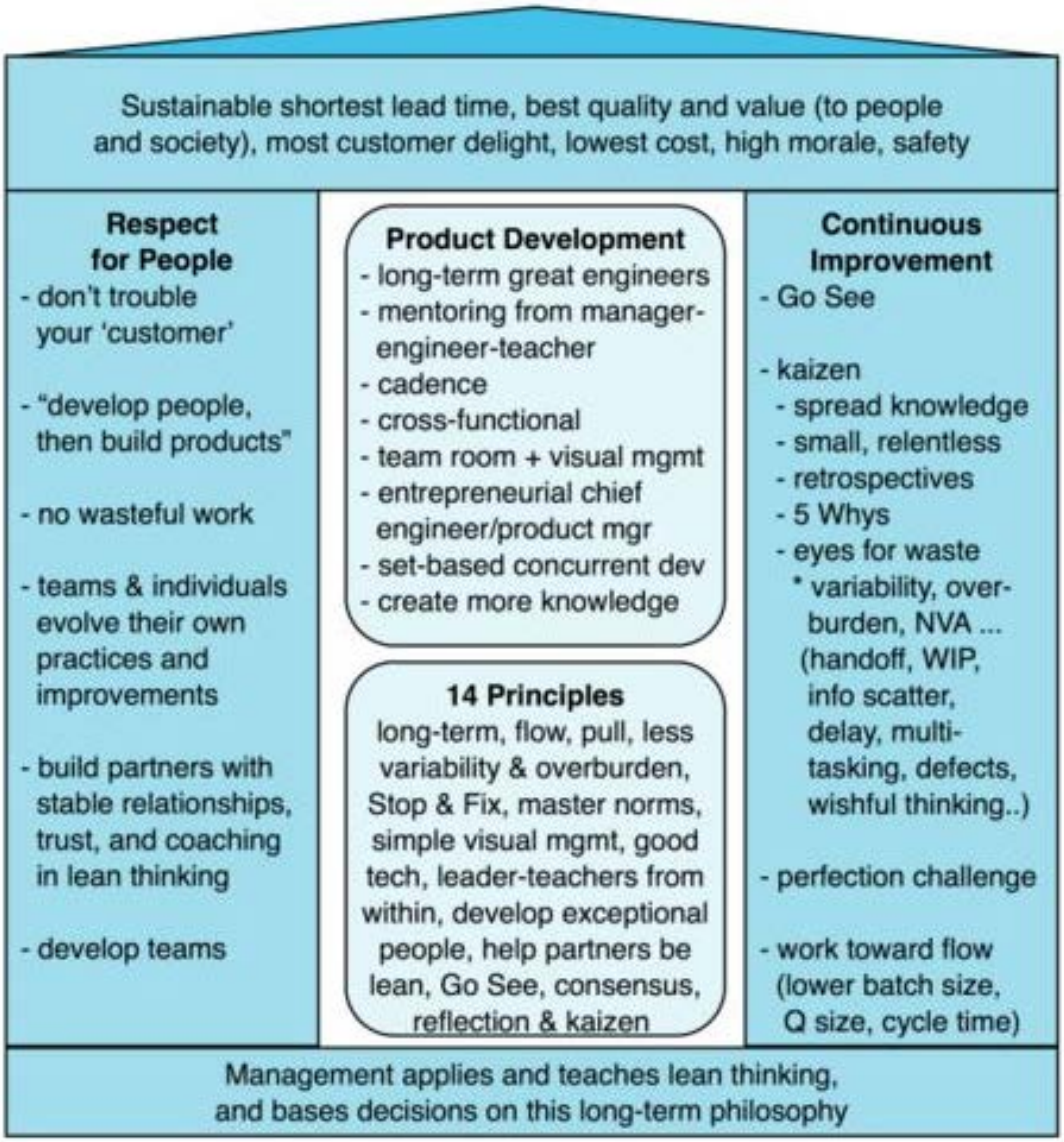


Figure 2.2.: Lean Thinking House by Larman and Vodde[30]

Lean House introduces concepts for the product development and 14 principles to follow when trying to work lean. Kaizen is a term used to describe the principle of continuous improvement, often associated with creating an environment where continuous learning and change are encouraged[33].

Differences between Lean Development and Agile Development

While there are many similarities between Lean and Agile Development and their core principles and values, they are not the same. Agile follows a more prototyping approach to product development of Do-Inspect-Adapt, while Lean follows a more organized approach of Inspect-Plan-Do [13]. Lean focuses on the enterprise value stream, while Agile Development focuses on value creation for the customer[13]. Hence, agile and lean have a different definition of what value is in software development[20]. Nonetheless, these two concepts are often combined in software development programs. According to Wang et al[63], there are six kinds of combinations for lean and agile practices:

1. Non-purposeful combination of agile and lean.
2. Lean for interaction with surrounding environment, agile within.
3. Lean for supporting agile adoption.
4. Lean to improve agile - Kaizen for continuous improvement.
5. Agile practices supporting lean processes.
6. Parallel and synchronized lean and agile.

With all the similarities, the practices of agile and processes of lean are often combined to reach a new level of improved process at an organization. Both concepts come with sets of principles aimed at improving the speed and quality of an product with customer needs and benefits in mind.

2.1.3. Scrum

Scrum is practiced at many organizations, hence, one of the most popular Agile Development frameworks[27]. Scrum as a framework was introduced by Ken Schwaber and Jeff Sutherland[48], who describe it as a 'framework for developing, delivering and sustaining complex products.'[48]. They go on to define Scrum as a framework allows people to deal with complex problems and still productively create and deliver products of high value[48]. All in all, Scrum consists of Scrum Teams with respective roles, artifacts and

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

Table 2.1.: Scrum Roles, Artifacts and Events adopted from [48]

events, see Table2.1, all aimed at supporting a development process. Scrum stems from empirical process control theory, stating that knowledge comes from experience[48]. To support this experience based process of decision making and development 'Scrum employs an iterative, incremental approach to optimize predictability and control risk'[48]. The responsibilities of the roles are clearly defined. The following roles, artifacts and events (Adapted from Schwaber and Sutherland[48], Wirdemann and Mainusch[66] and the Scrum Pattern Community[50]) where created to support the development process:

- **Roles:**
  Scrum Masters are responsible for dealing with impediments to the Scrum Team and promote Scrum both within the Team and to the outside.
  Product Owners are responsible for clear expression of the Backlog Items, prioritizing the Backlog, making all Backlog items visible, transparent and so every team member understands them.
  The *Development Team* is responsible for implementing the items of the Backlog in batches called Sprint Backlog during the Sprint. The end product of a Sprint is a Product Increment of a product, which can be delivered to the customer.

- **Artifacts:**
  Product Backlog is the collection of all requirements created by the Product Owner in cooperation with the customer. The Product Backlog is prioritized and estimated, so all requirements within the Backlog can easily be assigned to a Sprint.
  Sprint Backlog is the collection of all requirements, which will be developed during a Sprint. The Sprint Backlog is created by the Scrum Team by committing or denying the requirements proposed by the Product Owner for a Sprint.
  Increment is the result of all Sprint Backlog items, it is defined as a Product Increment which is 'Done' . The Product Owner decides whether it is released or not.

- **Events:**
  Sprint is defined as an iteration, time-period, during which a Product Increment is created.
  Sprint Planning is the event during which the Product Owner proposes Backlog items to be added to the Sprint Backlog. The Development Team discusses the items:
  1. What will be done during the Sprint?
  2. How to implement the Backlog Items which are done during the Sprint?
  3. Negotiate amount of Backlog Items committed to Sprint with Product Owner.
  4. Agree on a Sprint Goal.

  Daily Scrum is the daily meeting to communicate which work was done the last day and which work will be done the next day. Each member of the Development Team

presents his situation and mentions whether he has any impediments.
Sprint Review is the presentation of the created Increment during the Sprint from the Scrum Team to the key stakeholders.
Sprint Retrospective is used to critically inspect the Sprint, what was successful and more importantly what was unsuccessful and needs to be addressed.

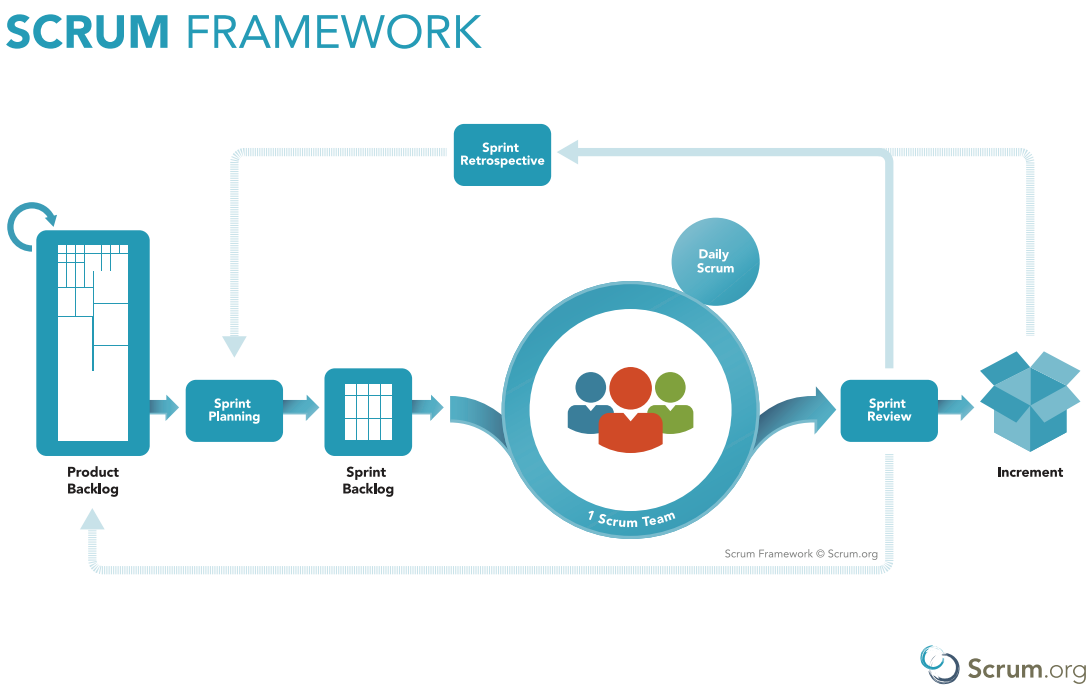


Figure 2.3.: Scrum Framework by Scrum.org[49]

The graphical depiction of the Scrum Framework by Scrum.org[49] shows the Scrum process as a whole. Overall, the Scrum is used as basis for many LSAD frameworks, which aim at scaling processes of the Scrum Framework (see Figure2.3) to fit for larger organizations, teams and projects.

2.2. Large-Scale Agile Development

This section will introduce the term LSAD and defines what can be understood as a LSAD. Afterwards, two frameworks will be introduced, the Large-Scale Scrum (LeSS) in Section2.2.2 and the Spotify model in Section2.2.3. In the end of this section is an overview of further LSAD frameworks in Section2.2.4.

2.2.1. Definition

LSAD in its core describes different methods and concepts on how to scale agile practices. The need to scale originates in agile practices, as these were originally designed for small projects. William and Cockburn state". . . the agile value set and practices best suit co-located teams of about 50 people or fewer who have easy access to user and business experts and are developing projects that are not life-critical. "[65]. However, the benefits associated with agile practices, have led to high interest from large organizations. In general there is some ambiguity about what defines LSAD[21].
Dingsøyr et al.[19] define a taxonomy for LSAD projects, based on a similar taxonomy in requirements engineering. Therefore, they define large-scale as any project with two to nine teams.
Dikert et al.[18] define large-scale as any project with more than 50 people and at least six teams.
These definitions of large-scale all associate the amount of teams and people involved in the project as driving factors to defining what is and what is not a large-scale project. In their work to define principles for LSAD, Dingsøyr and Moe[22], go on to prove that people working with agile practices have a similar understanding for the term large-scale. During a workshop, Dingsøyr and Moe[22] identified different understandings of the term large-scale, with size of teams and amount of people involved at the core of most definitions. Rolland et al.[44] identified that'...,dealing with an increasing number of actors, interfaces with existing systems, and unexpected interdependencies – this is what distinguishes large-scale projects from traditional ones'[44] is what makes the difference between agile development and LSAD. Bringing together in-house, root metaphor, paradigmatic,ideological and field assumptions to find a definition for agile in the large[44]. This thesis applies aspects of the definition of Power[43]. Power[43] identified three aspects to working in scaled agile organizations:**Agile approaches in large organizations**, **Agile approaches in a large development effort in a large organization** and **Organizational agility**. The second aspect implies a need to scale agile methods to support the approach of working agile in a large development effort. As well as, the interdependency and intercommunication aspect from Rolland et al.[44]. Combined with Dingsør et al.[20] taxonomy of large-scale, this thesis's defines LSAD as follows:

Large-Scale Agile Development utilizes agile methods, principles and values in two to nine agile teams. These agile teams are inter-dependent and work together on a project.

Agile approaches applicable to both co-located or geographically distributed teams. Agile teams working together on a project need to communicate and coordinate on many topics, for example 'Who is responsible for the development of which backlog items?', 'How can development on modules be synchronized?' and many more challenges.

2.2.2. Large Scale Scrum

Large Scale Scrum (LeSS) defines a framework for scaling agile methods associated with Scrum. LeSS was designed by Larman and Vodde, who describe it as "Scrum applied to many teams working together on one product"[31]. With their organization TheLeSSCompany B. V. they define two frameworks, LeSS and LeSS Huge. The difference between these frameworks lies in the amount of teams working together. LeSS is applied for up to eight teams and LeSS Huge for more than eight teams and up to a few thousand people working on the same product[4]. LeSS relies on the core principles of Scrum. It can

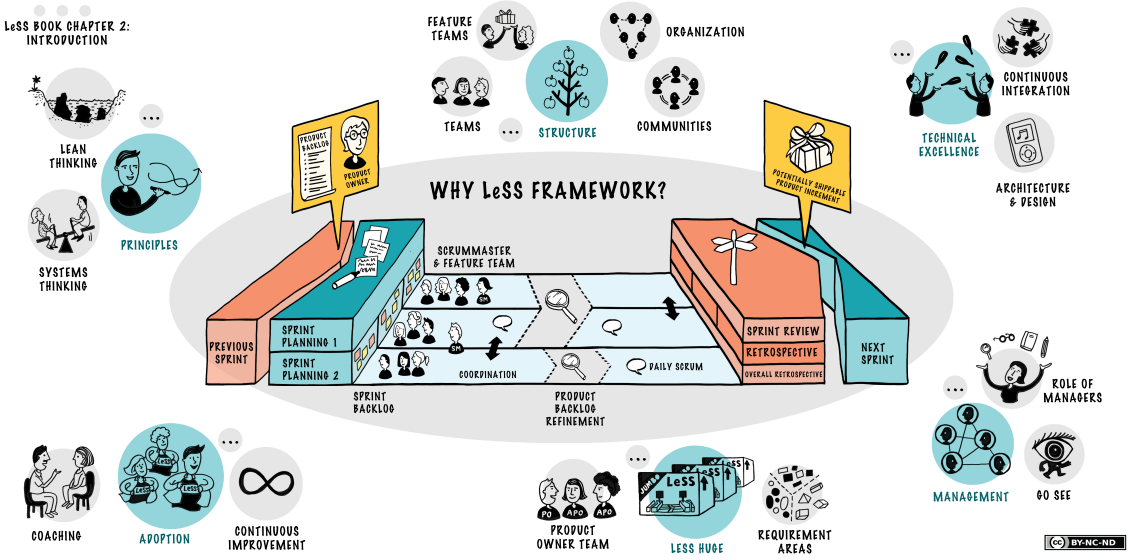


Figure 2.4.: Large-Scale Scrum Framework[3]

be seen as several individual Scrums working together with the additional concepts of theLeSSframework. Therefore, at the foundation of everyLeSSinstance is the team working in a Scrum process, see Figure2.4. All Teams share **one Product Backlog**, **one Product Owner** and they work on **one Increment**. However, each Team consists of its own Scrum Master and Development Team. The teams work in the same sprint, so they have synchronized sprints, leading to Sprint Planning, Review and Retrospective occurring at all teams at the same time [31]. Sprint Planning is split up into two instances, Sprint Planning One and Sprint Planning Two. Sprint Planning One is used to assign requirements from the Product Backlog to teams. In Sprint Planning One the teams will send representatives, who in cooperation with the Product Owner will work out the allocation. Sprint Planning Two is similar to Sprint Planning in Scrum, where the individual teams perform their Sprint Planning. In cases where backlog items allocated to two or more teams are connected, Sprint Planning Two can be performed together for those teams[31]. Similar to Scrum, the Sprint is closed by the Review, Retrospective and Overall Retrospective. Continuous Im-

provement is one major principle ofLeSSand the emphasis of Larman and Vodde[31] on an Overall Retrospective, discussing cross-team impediments and issues during the sprint to improve the following sprints, is reflected by that. The Overall Retrospective is attended by the Product Owner, Scrum Masters and team representatives, in some cases managers attend as well. The importance of the Overall Retrospective is reflected in the importance of the Team Retrospective or for simplicity Retrospective. In a Retrospective the same discussion is held on a team level, with all team members attending. The Sprint Review is used for the teams to present their results of the Sprint to the Product Owner and eventually attending Customers[31].
One event to draw special focus on is the Product Backlog Refinement(PBR), it occurs during the Sprint and aims at refining items still in the Backlog. Product Owner and the Sprint Teams are part of this event and they work together to define items to the stage of ready and estimate the items, to allow future allocation to sprint to be more accurate. LeSS defines three types of PBR, (1) Multi-Team (2) Overall and (3) Single-Team. The order of the mentioned PBRs represent the usefulness associated with them by The Less Company B.V.[4]. LeSS clearly states that doing a Mulit-Team PBR is the most useful, while they would avoid doing a Single-Team PBR as it reinstates boundaries and weakens coordination and alignment[4]. LeSS and LeSS Huge employ Feature Teams, which are cross-functional and cross-component full stack teams[30]. A Feature Team is able to cover all aspects of development and consists of five to nine generalizing specialists[30].
Figure2.4 consists of all relevant artifacts, meetings and roles of the LeSS Framework. LeSS Huge can be seen as several instances of LeSS stacked on each other[4, 31]. LeSS Huge also only uses one Product Backlog and a synchronized Sprint, however, a team of Product Owners is employed to deal with the larger setting. The setup consist of so-called Requirements Areas(RA)[31]. These RA are managed by an *Area Product Owner*, who is assigned by the Product Owner. Each RA consists of four to eight *Area Feature Teams*, who work with the Area Product Owner. The Product Owner focuses on the complete product development, while the Area Product Owners focus on their specific area product development. The Product Owner is responsible for allocating the Area Product Owners and splitting up the Product into RA[31].
In Combination with the proposed framework, LeSS relies on a set of principles visualized in Figure2.5. These principles (see Figure2.5) stand for:

- **Large-Scale Scrum is Scrum** - LeSS is Scrum scaled on all levels instead of applying it on team level and adding a scaling process[4].
- **More with LeSS** - LeSS aims at reducing organizational complexity and solve product development in a simpler manner[4].
- **Systems Thinking** - Set of tools to see systems dynamics, mental models and local optimization[30].
- **Lean Thinking** - Build people and challenge the status quo, creating a mindset for continuous improvement[26].

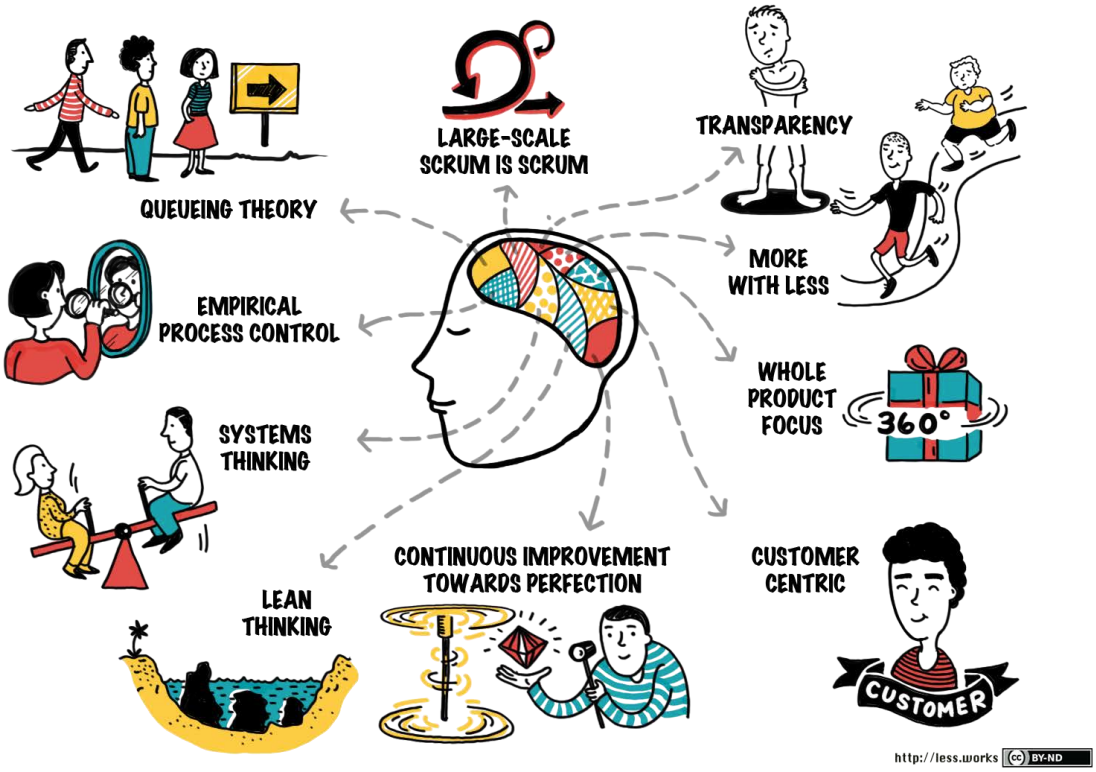


Figure 2.5.: LeSS Principles from The LeSS Company B. V.[3]

- **Empirical Process Control** - Creating increments of the product, which can be shipped and inspected for future adaptions of the process[4].
- **Transparency** - To be able to inspect and adapt your process and enable continuous improvement.
- **Continuous Improvement Towards Perfection** - there is no final state of the product or process, challenge the status quo and try to achieve a better product/process every day.
- **Customer Centric** - The customer is the core of the product, focus on creating value to the customer.
- **Whole Product Focus** - Focus on the whole product not on the individual parts.
- **Queueing Theory** - Queues affect the cycle time, so aim at eradicating Queues, by visualizing the remaining queues and limiting queue sizes[30].

2.2.3. Spotify

The Spotify model introduced in 2012 by Kniberg and Ivarsson[28] works with a scaling model comparable to matrix organizations, introducing four units of working groups,**Squads**, **Tribes**, **Chapters** and **Guilds**, see Figure2.6.

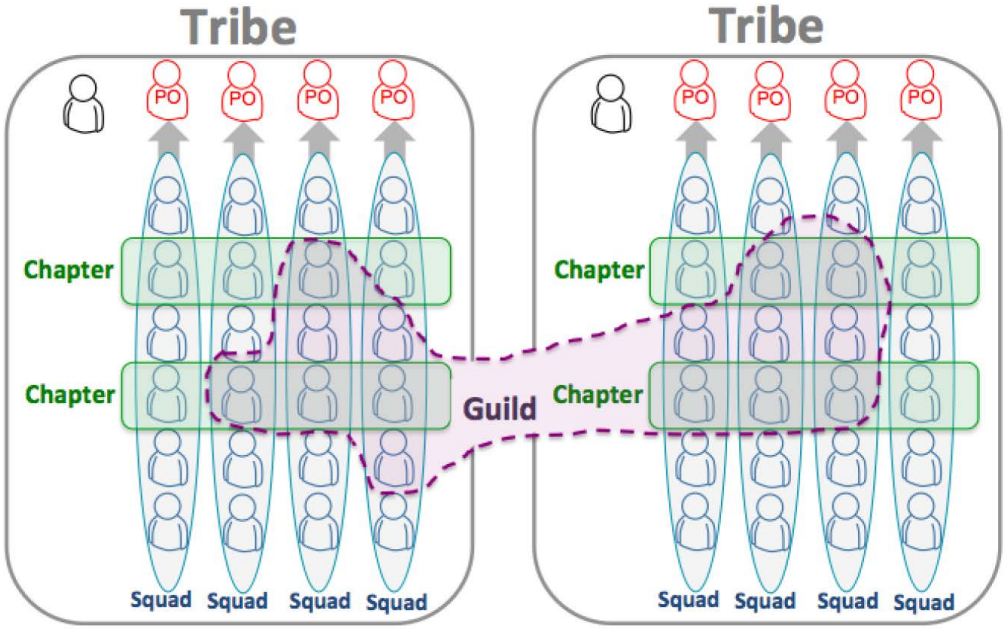


Figure 2.6.: Spotify Model from Kniberg and Ivarsson[28]

The *Squad* represents the basic unit of development[28] and is similar to a Scrum team. A Squads' Product Owner is responsible to further the product development and prioritize the backlog items. One Squad holds all capabilities required to deliver a product increment, comparable to Feature Teams except for having an Agile Coach instead of a Scrum Master to help identify and solve impediments. Squads are meant to be autonomous, however, dependencies between squads exist and they are represented in a dependency excel to help resolve dependencies and make them visual[28]. Several Squads working in a similar area make up a *Tribe*. Tribes are usually co-located, meaning that all Squads of a Tribe have it easier to co-operate with each other and exchange knowledge[28]. Within the Tribes, regular events comparable to the Sprint Review occur, to showcase the work done by individual Squads. Facing the issue of Economies of Scale, the Spotify model introduces Chapters and Guilds. Chapters and Guild offer a platform for employees working in the same role to exchange knowledge and solutions already designed. *Chapters* consist of people of the same roles in the same Tribe, with one Chapter Lead as line manager for his chapter members[28]. Whereas *Guilds* are comparable to Communities of Practice and spread across Tribes, similar to Communities of Practice, Guilds depend on a coordinating

figure to help guide the Guild[28]. Finally, Spotify introduces the concept of the Operations

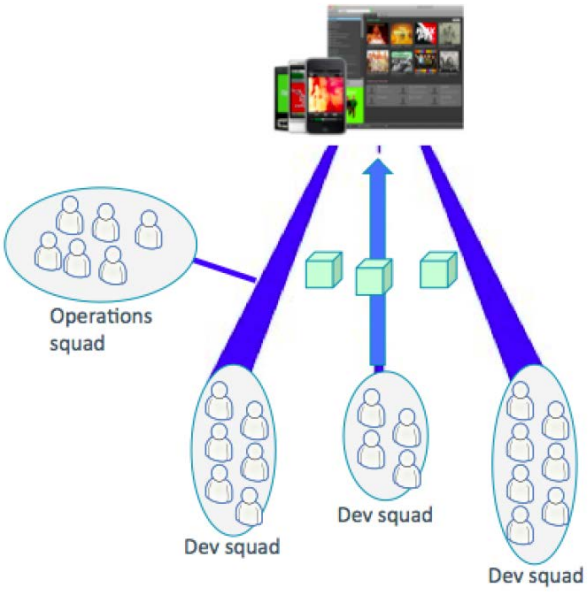


Figure 2.7.: Spotify Release Support through Operations Squad[28]

Squad, a Squad with the sole purpose to deliver the infrastructure for the other Squads to make releases on their own, see Figure2.7.

2.2.4. Overview of Further Large-Scale Agile Development Frameworks

For this thesis the above mentioned agile practices (Scrum) and scaled-agile frameworks(LeSS and Spotify framework) are most relevant as the case observed in this thesis adopted their LSADP based on concepts from these two frameworks. However, there are many more agile practices and LSAD frameworks, for example the Scaled Agile Framework(SAFe®) which is applicable from a few teams to a whole organization, four-level SAFe®[36]. While SAFe®finds application at the case study partner, the LSADP observed in this thesis's case intentionally did not apply SAFe®, the reasons for this decision are described in Section4.2. Additional frameworks are the Scrum-at-Scale and Scrum-of-Scrum[53] and many more, which are summarized and explained at [61].

2.3. Pattern Language

This section introduces the Pattern Language created by Uludağ et al.[55], which is based on the Pattern-based Design Research by Buckl et al.[10]. However, we will begin with the definition and terminology of Pattern in Section2.3.1. Afterwards, we will introduce

some Software Development Pattern Languages in Section2.3.2. Finally we'll introduce the LSADPL, in Section2.3.3, applied in this thesis.

2.3.1. Definition Pattern

The definition for a pattern, "Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution. "[2] is the basis for most work in the area of pattern research. **Context**, **Problem** and **Solution** are the three elements found in every definition of a pattern. Furthermore, patterns help documenting observations from practice in an organized manor, which makes them reusable[24]. Buschmann et al. [11] state that patterns endorse good practice, by documenting "existing, well-proven experience". Similar to those three elements it is defined by Coplien[15] that a best practice has to be implemented at least three times in similar fashion, for it to be regarded as a pattern. This *Rule of Three* is applied in this thesis to differentiate between patterns, which have to be observed at least three times and Pattern Candidates, which were observed less than three times. By this *Rule of Three* a distinction is made for this thesis between best practices, a pattern observed three or more times in similar implementations and a good practice a single occurrence of an observed Pattern Candidate. When several patterns are combined, a Pattern Catalog or Pattern Language is created. For this thesis, the LSADPL by Uludağ et al.[55] is applied (see Section2.3.3). A Pattern Language offers a connection between multiple Patterns in a meaningful way[14].

2.3.2. Software Development Pattern Languages

Pattern Languages as a collection of patterns in one topic are help address problems and resolve them by applying observed solutions. In the area of Software Development, Scrum can be seen as a collection of patterns to address problems arising in the agile development progress. As such Beedle et al.[8] defined the Scrum Pattern Language, which build the patterns by describing Context, Problem, Forces, Solution, Rationale, Examples and Resulting Context for the artifacts and events of Scrum, compare Section2.1.3. Building patterns following the concepts of Context, Problem and Solution originates from an observation by Christopher Alexander, mentioned in an introduction by James O. Coplien[1]. The combination of these building blocks of a Pattern Language can also be observed in the conceptual overview (see Figure2.8) ot the Enterprise Architecture Management (EAM) Pattern Language by Schneider and Matthes[47]. The evolved EAM Pattern Language of Schneider and Matthes[47] is based on the EAM by Buckl et al.[10] and introduces the relation between the different concepts **Influence Factors**, **Stakeholders**, **Concerns**, **Method Patterns & Principles**, **Viewpoint Patterns** , **Information Model Patterns** and **Data Collection Patterns**, compare Figure2.8). Similar to Scrum the relationships between the different concepts are used to describe a pattern for an EAM solution. Again building the pattern structure by using the building blocks, defined by Schneider and Matthes[47], **Context**, **Problem**, **Solution**, **Forces**, **Consequences**, **Stakeholders** and **See Also**. In a

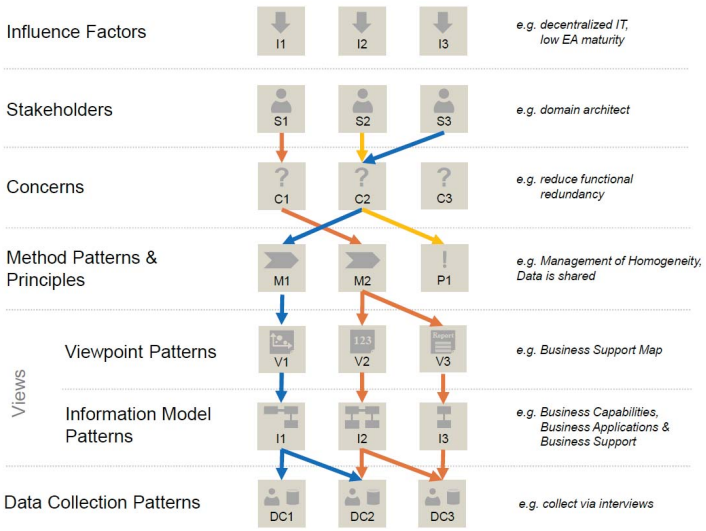


Figure 2.8.: Conceptual overview Enterprise Architecture Management Pattern Language by Schneider and Matthes[47]

similar template of Context, Problem and Solution relation, as well as supporting building blocks for a detailed description of the pattern, are all Pattern Languages build.

2.3.3. Large-Scale Agile Development Pattern Language

Pattern Languages aim at combining patterns observed in a similar area of research. The LSADPL applied in this thesis was designed by Uludağ et al.[55] and founded on the Enterprise Architecture Pattern Language by Buckl et al.[10]. Following the work of Uludağ et al.[55], this thesis uses the LSADPL, which combines five concepts (compare Figure2.10) **Stakeholder**, **Concern**, **Large-Scale Agile Pattern**, **Large-Scale Agile Development Anti-Pattern** and **Principle**.
The conceptual overview of the LSADPL can be seen in Figure2.9
The definitions of these concepts are shown in Table2.2.

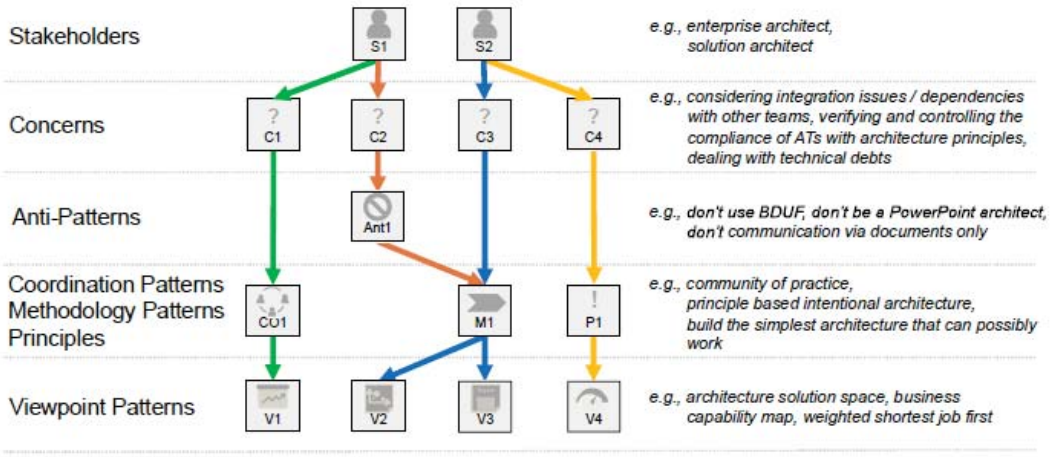


Figure 2.9.: Conceptual overview Large-Scale Agile Pattern Language by Uludağ et al.[55]

The Figure2.10 outlines all concepts, their types, elements and relation in detail.
For this thesis, the Pattern Language introduced and the definitions for this Pattern Language as well as the general definitions regarding practices, patterns and pattern candidates from Section2.3.1 apply. The patterns of the LSADPL by Uludağ et al.[55] consist of the following building blocks, here the Template for a Coordination Pattern is presented as designed by Uludağ et al.[55]:

- **Pattern Overview**- general Information as the Name and a Summary of the Pattern.
- **Example** - Correct application of the pattern.
- **Context** - Circumstances creating the Problem.
- **Problem** - Problems addressed by pattern.
- **Forces** - Reasons why the problem is difficult to resolve.

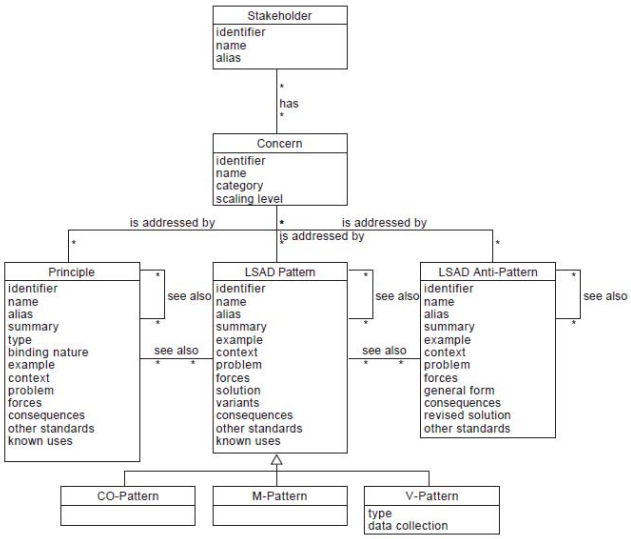


Figure 2.10.: Pattern Language Structure by Uludağ et al.[55]

- **Solution** - Description of how the problem is solved.
- **Variants** - Different solution variations for the problem.
- **Consequences** - Benefits and Liabilities of pattern.
- **See Also** - Is the pattern applied in combination with other patterns.
- **Other Standards** - Do other Frameworks advice the use of the pattern.
- **Known Uses** - Which organizations apply the pattern.

This pattern template finds application in this thesis for documenting pattern candidates observed at the case study partners' LSADP.

Patterns and Concepts	Definition
Stakeholder	Individual or group of individuals Involved or affected by LSAD[57].
Concern	Responsibility,task, challenge or goal of an stakeholder. Concerns reference principles and patterns aimed at resolving these concerns[55].
Coordination Pattern(CO-Patterns)	CO-Patterns describe coordination processes(e.g. tasks) to address recurring concerns in LSAD.
Methodology Pattern (M-Patterns)	M-Patterns define concrete steps for addressing concerns in LSAD.
Viewpoint Pattern (V-Patterns)	V-Patterns offer ways of visualizing information(boards, documents, metrics, models and reports) addressing recurring concerns in LSAD.
Principle	Principles consist of rules and general guidelines, they might help address recurring concerns in LSAD.
Anti-Pattern	Anti-Patterns describe common mistakes in LSAD and revised solutions to avoid these mistakes.

Table 2.2.: Patterns and Concepts following the definition of Uludağ et al.[55]

2.3.4. Stakeholders

This Section introduces the stakeholders groups Scrum Master (SM) (Section2.3.4), Product Owner (PO) (Section2.3.4) and Development Team (DEV) (Section2.3.4) which are the focus of this thesis. These stakeholder groups and 11 more were identified in Uludağ et al. [57] with a literature review on identifying and structuring challenges in LSAD. The LSADPL created by Uludağ et al.[55] goes on to combine these stakeholder groups with their challenges.

Development Team

Responsibilities
The Development Team(DEV) is responsible for the development of the Backlog Items assigned to a Sprint and delivering an Increment of the product as defined in Section2.1.3. DEV is the combination of all team members working on the development of requirements, in some cases this can include the SM in a 50 % part-time role. Additionally, DEV is responsible for clarifying non-functional requirements and during the PBR help the POestimate User Stories accurately. The DEV is self-organizing and usually presents the Increment during the Sprint Review to the customer and POand in some scaled-agile they are also responsible for releasing these Increments[28].
Challenges
For DEV, Uludağ et al. [57] identified four challenges:

- How to establish self-organization?
- How to create lightweight documentation?
- How to explain requirements to stakeholders?
- How to coordinate tests and deployment with external parties?

None of these challenges have a direct correlation to LSAD.

Product Owner

Responsibilities
The Product Owner (PO) is responsible for the overall development of the product. Prioritization of items in the Backlog is the major task for the POas he is responsible for identifying which items will bring the biggest value to the customer and therefore should be addressed first[31]. POs are responsible for clarifying functional requirements with the customers of the product and are the interface between agile team and customer[35, 48]. As such, the PO is also responsible for eliciting requirements and transforming them into User Stories[66] and in a final step he refines these User Stories in cooperation with the Development Team[48]. In the end, the PO decides when to release an Increment to customers.

Challenges
Most of the challenges associated with the PO are connected to Requirements Engineering, some challenges are connected to Project Management. For the POstakeholder, Uludağ et al. [57] identified a total of 14 challenges:

- How to create precise requirement specifications for the development team?
- How to share a common vision?
- How to elicit and refine requirements of end users?
- How to split large and complex requirements into smaller requirements?
- How to facilitate communication between agile teams and other teams using traditional practices?
- How to communicate business requirements to development teams?
- How to define clear and visible priorities?
- How to deal with unplanned requirements and risks?
- How to enforce customer involvement?
- How to make a cost and schedule estimation?
- How to create and estimate user stories?
- How to establish requirements verification?
- How to define high-level requirements a.k.a. epics?
- How to measure the success of the large-scale agile development program?

These challenges are not limited to LSAD, but can also be observed in small agile projects.

Scrum Master

Responsibilities
The Scrum Master (SM) is as a part of the agile team responsible for enabling the development process. In his role he helps the DEV in identifying and resolving impediments arising during the development process. Overall, the SM is responsible for productivity and an efficient Sprint[66]. As a enabler he supports not only the DEV in their endeavors, but also the Product Owner. In a scaled agile approach, he is further needed as a scrum of scrum facilitator for communication and coordination between the different agile teams[6]. As introduced in Section2.1.3 the SM is responsible for advocating and promoting agile practices not only inside the team, but also to the outside. Additionally, the SM

is responsible for moderating and managing the meetings, making sure that the meetings fulfill their purpose as defined by him[48]. In the practice, the SM is rarely represented by a full time role, many agile teams use the SM as a 50% role often in combination with a 50% developer[17, 38]. A part-time SM is often more respected by the Development Team, as he is regarded as more experienced and more understanding of impediments facing developers[17].

Challenges

The SM faces multiple challenges traditionally associated with project managers' concerns. However, he additionally has to face several challenges regarding agile practices and supporting the agile process. The challenges identified by Uludağ et al. [57] for the SM are mainly aimed at issues concerning geographical distribution. In total, ten challenges were identified in this literature review:

- How to coordinate geographically distributed agile teams?
- How to facilitate agile teams to participate at cross-shore meetings?
- How to synchronize working hours of cross-shore agile teams?
- How to deal with lacking team cohesion at different locations?
- How to build trust of stakeholders in agile pratices?
- How to create a culture of continuous improvement?
- How to rearrange physical spaces?
- How to deal with higher-level management interference?
- How to deal with cultural differences between cross-shore agile teams?
- How to synchronize sprints in the large-scale agile development?

Many of these challenges do not only apply to large-scale, but can also be found in small agile projects.

3. Related Work

In this chapter we will introduce some related work in the area of LSAD in Section3.1, work on identifying recurring concerns in LSAD in Section3.2 and finally introduce related work observing the learning process of agile programs in Section4.5.

3.1. Related Work on Large-Scale Agile Development

Fuchs and Hess (2018) - Becoming agile in the digital transformation: the process of a large-scale agile transformation
Fuchs and Hess[25] inspect the process of agile transformation with a multiple-case study design. As a Result, Fuchs and Hess[25] identify different agile phases of transformation, each distinguished by organizational efforts at the beginning of each phase. The phases are **Initial Transition to agile practices**, **Coping actions towards challenges of initial agile phase**, **Addressing challenges of second agile phase** [25]. Overall, observing the wave-like process of agile transformation, Fuchs and Hess[25] were able to distinguish the different phases through the organizational effort that initiates each phase, interrupting the flow with radical change[25]. Instead of an incremental transformation, agile transformation was found to need several boosts, to initiate new levels of organizational agility[25].

Paasivaaraet al. (2018) - Large-scale agile transformation at Ericsson: a case study
Paasivaara et al.[39] identified with their case study four phases of agile transformation. **Knowledge Transfer and Component-Based Teams**, **Introducing Agile**, **Finding Common Ground Through Value Workshops** and **Towards Continuous Integration and Deployment**[39]. Each phase is distinguished by increased efforts to achieve a more agile organization. However, during the phases, they identified inherent challenges, which challenged the use of , e.g. cross-functional teams and co-located teams[39]. Ericcson reacted by transforming into their own setup, focusing on creating common values with value workshops and including teams as soon as the design and planning for functionalities[39]. The challenges observed by Paasivaara et al.[39] range from **Change Resistance**, **Technical Debt**, **Cross-Site Teams** to **Specialized Teams** to name a few. Leading to the Lessons Learned of which two are of increased interest for this thesis, **Limited Team Interchangeability** teams working on business flows, of which they are experts [39]. Secondly **Stepwise Transformation** the agile transformation is not a single big bang process as organizations need to continuously deliver features, it is more of an step-wise change[39].

Dingsør and Moe(2014) - Towards Principles of large-scale agile development The work of Dingsør and Moe[22] identifies four principle categories for LSAD.

1. Architecture principles - Coordinating work and the influence of the level of change on the architecture organization.
2. Inter-team coordination principles - norms and values facilitate communication and the need for effective knowledge networks in LSAD.
3. Portfolio management principle - consistent alignment between teams and portfolio vision to achieve the overall portfolio goals and feedback from the LSAD teams to the portfolio to optimize value of LSAD.
4. Scaling principles - context of agility and scale essential for improving agility and agility scaled with respect to involved members.

These principles aim at making LSAD success more achievable, by focusing on the correct adaption of LSAD frameworks.

Uludağ et al.(2019) - Using Social Network Analysis to Investigate the Collaboration Between Architects and Agile Teams: A Case Study of a Large-Scale Agile Development Program in a German Consumer Electronics Company
Uludağ et al.[58] observe the collaboration of architects with agile teams and identify methods of interaction, as well as how they are perceived and which benefits stem from these methods. One Finding is the preference of direct communication by the architects, for which two types of communication were observed inter-team and intra-team communication. Observing the affiliation between Architects and squads and, to what extent and how often communication happens. To generate a detailed case description and document the adopted agile approach Uludağ et al.[58] uses a mixed method approach to identify and described the agile program observed. Creating a LSADP framework of a German consumer electronics retailers' agile release train, though a combination of literature review, observations from events and experts interview. All these data inputs are aimed at eliciting certain key aspects of the case study partners' LSADP. Through the social network analysis, Uludağ et al.[58] could identify the Solution Architects as central communication nodes. Indicating their high communication efforts and involvement with the agile teams. A similar approach finds application in this thesis, with the foundation of 2.2.2 and 2.2.3 and interviews with experts at the case study partner to elicit the LSADP of the case study partner.

3.2. Related Work on Identifying Recurring Concerns in Large-Scale Agile Development

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

The work of Uludağ et al.[57] creates a basis for researching LSAD as they introduce not only challenges observed, but also stakeholder groups and categories for the observed challenges. The concept of recurring concerns as well as the identification of different stakeholder groups created the foundation of this thesis. The research paper goes on to identify 79 challenges of fourteen stakeholder groups by conducting a literature review over 73 relevant sources. Challenges are categorized into one of eleven categories: **Communication & Coordination**, **Software-Architecture**, **Geographical Distribution**, **Knowledge Management**, **Methodology**, **Culture & Mindset**, **Tooling**, **Requirements Engineering** and **Project Management**. Based on the work of Cruzes and Dybøa[16] and Ernst[24] they go on to identify the concepts and pattern types:**stakeholders**, **challenges**, **methodology patterns**, **architecture principles**, **viewpoint principles** and **anti-patterns**. The challenges are then mapped to the stakeholders and in a novelty check, controlling, whether a challenge is relevant for LSAD or can also be found in normal agile projects. Further, the work identifies that not all challenges of LSAD are documented in the literature, there are still challenges that need to be identified. With this work, the research project this thesis is a part of is initiated, which later produces the LSADPL[55], which is applied in this work.

Dikert et al.(2016) - Challenges and success factors for large-scale agile transformations: A systematic literature review
Compared to Uludağet al.[57], the work of Dikert et al.[18] identifies 29 success factors in addition to 35 challenges in agile transformations. The work researches agile transformations utilizing a literature review of 52 papers, including experience reports[18]. Dikert et al.[18] identify nine challenge categories from the literature review with 'Agile being difficult to implement' being mentioned by 48% of the cases observed. Similarly often identified where concerns of the category 'Integrating non-development functions' at 43%. The most often observed success factors were 'Choosing and customizing the agile approach' with 50%, 'Management support' and 'Mindset and Alignment' both with 40%. From these findings, Dikert et al.[18] concluded that for an agile transformation to be successful, the framework has to be adopted to the organization and without addressing the values behind agile, a transformation is less likely to be successful.

3.3. Related Work for identifying good practices in Large-Scale Agile Development Programs

Paasivaara, M. (2014) - Communities of practice in a large distributed agile software development organization–Case Ericsson In this work Paasivaara and Lassensius[40] go on to identify eight characteristics of a successful Community of Practice(CoP). The eight characteristics were identified by observing different Community of Practice (CoP) (see Figure3.1) and collecting experiences through expert interviews.

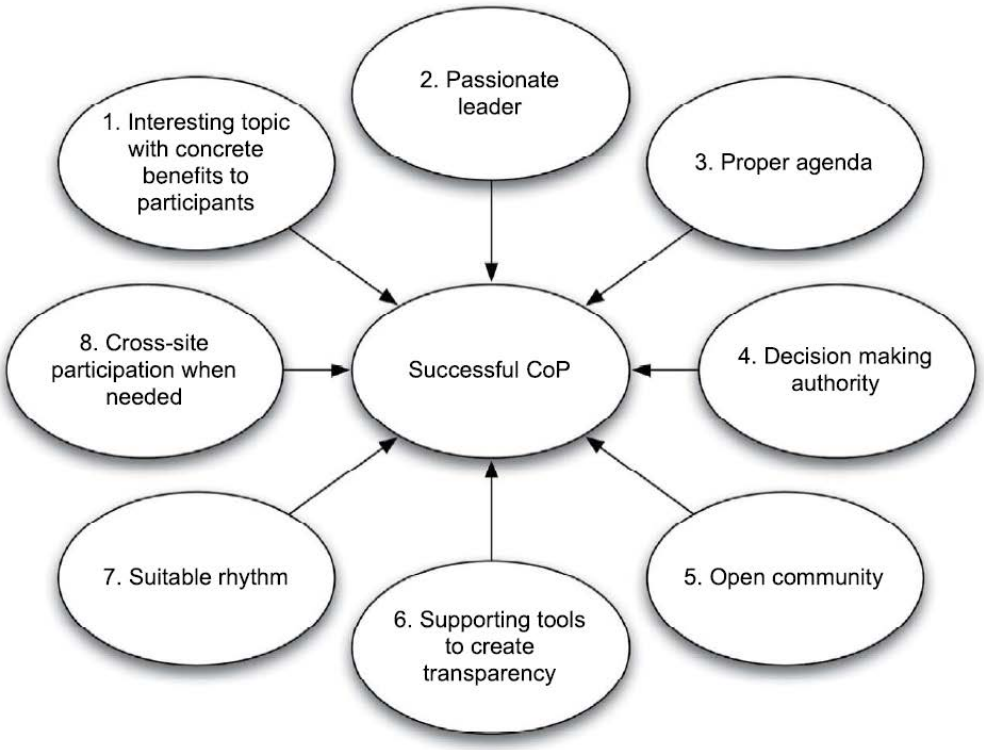


Figure 3.1.: Characteristics of a successful Community of Practice[40]

Uludağ et al.(2018) - Supporting large-scale agile development with domain-driven design
The effect of architects and the adoption of (Domain-Driven-Design (DDD) is the foundation of this work by Uludağ et al.[56]. The work not only describes the LSADP of the case study partner, but also introduces methods applied, which can be seen as best practices of applying DDD at the case organization. With the tactical DDD approach using the Event Storming process, which generated a a comprehensive model of the business flow[56], this thesis has a pattern, which is introduced to this thesis' case study partner as a possible solution for several concerns. Uludağ et al.[56] finds that to adopt DDD in a LSAD, organizations do not need to add further roles. However, the responsibilities of the roles have to updated. Development teams need to develop and maintain domain models to help the strategical DDD and the importance of architects increases. Overall, the observed framework is a combination of the tactical and strategical DDD provided and maintained by the teams and architects surrounding a LeSS approach. For this thesis, the concept surrounding the event storming workshop is focus point as it allows architects and the team to align and conceptualize the domain architecture and helps to generating a better understanding of a module and all occurring events [62].

Uludağ et al. (2019) - Documenting recurring concerns and patterns in large-scale agile development
In Addition to the LSADPL described in Section2.3.3. The work of Uludağ et al.[55] documented patterns as well as mapped these patterns to concerns. Making it easy to search for possible solution designs by focusing on the concerns observed in a LSADP. The patterns documented by Uludağ et al.[55] are available with the introduced pattern language. Uludağ et al.[55] go on to document the currant concerns, patterns and anti-patterns, which are part of the pattern language designed. Two of these patterns **CO-1** and **CO-2** are introduced to the case study partner of this thesis. Additionally, the pattern language allows documentation of practice-proven solutions to recurring concerns[55]. This thesis goes on to work on documenting best practices, recurring concerns and applying the aforementioned pattern of **CO-1** at the case study partner. Essentially, working on the future work described by Uludağ et al.[55]. Additionally the star-notation representing the confidence level used in this thesis was introduced by Uludağ et al.[55]. The maturity of a pattern is represented by star notation: two stars, indicating the pattern addresses a genuine problem, one star indicates the pattern addresses a real problem, no star means the pattern was useful for a observed problem but needs revision[55].

4. Case Study

This chapter summarizes all Findings observed during the case study research. First, the case study partner will be introduced and a case description will be offered in Section4.1. Next, this thesis will introduce Then this thesis will introduce the LSADP Framework currently in use at the case study partner in Section4.2. The findings of the second interview round with recurring concerns and good and bad practices is discussed in Section4.3. Followed by identifying recurring concerns for the three stakeholder groups in the case study in Section4.3.2. Section 4.4 will introduce Good and Bad Practices observed in the case study and finally the thesis will close the findings with the implementation of provided Patterns and the observed Lessons Learned in Section4.5.

4.1. Case Study Description

This section introduces the case study partner, Technology LLC., and describes their current team situation(Section4.1.1) of the LSADP. The interview process and interview partners are introduced in Section4.1.2 and the findings of these interviews are then presented in Section4.2, Section4.3 and Section4.4.

4.1.1. Team and Platform of Technology LLC.

The case study partner of this thesis is situated in the technology sector, therefore we will call it Technology LLC..

Role	Team A	Team B	Team C	Team D	Team P	Total
Scrum Master	0.5	0.25	1	0.25	2	4
Product Owner	3.08	3.08	1.25	2.58	1	11
Test Team	2	4	1.5	1.5		9
Process Consultant	0.5	2	1	1		4.5
Development Team	2	3.25	3	1		11.5
IT-Architect					1	1
Software Architect					1	1
Test Architect					1	1
Sum of Members	8.08	12.58	7.75	8.58	6	43

Table 4.1.: Team conception at case study partner

Technology LLC. operates as an automobile supplier, invests in the internet-of-things and has been a leading producer of household appliances for a long time. While the organization itself has several projects working with the Scaled Agile Framework (SAFe®) framework, the observed LSADP at the case study partner decided in early 2017 to adopt the LeSS framework.

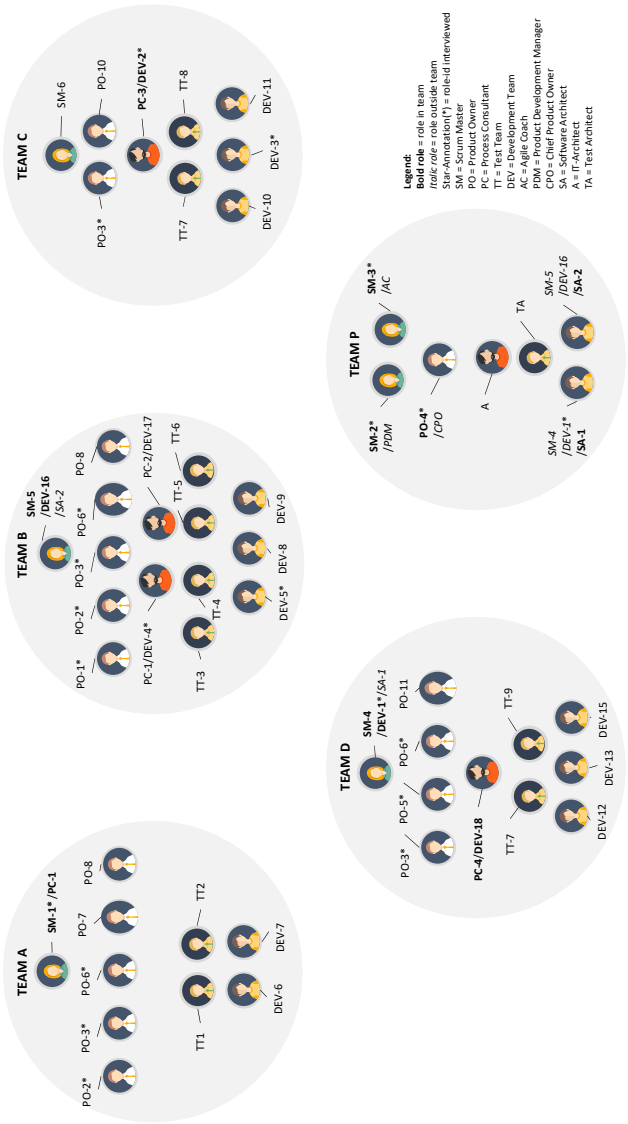


Figure 4.1.: Detailed view of each Team at case study partner

The teams consist of eight different stakeholder groups, three of which were the focus of the interviews for identifying recurring concerns, in Section4.3, and documenting good and bad practices ,in Section4.4. The teams are aggregated as depicted in Table4.1 and visualized in Figure4.1. The role Test Team is associated with manual testing capabilities. Automation testing is handled by the Test Architect (TA) in cooperation with each teams' Development Team. Process Consultants split their work into two dedicated 50 % blocks, one part developer and one part technical expert, responsible for supporting other team members with technical impediments. Finally, as observed in other cases, the Scrum Master role is only partly filled, as most Scrum Master fulfill another responsibility as Development Team member or in the case of SM-2 and SM-4 as 50 % Software Architects(SA) and 25% SM and 25% Development Team member[38].

Figure 4.2.: SunBurst of the teams distribution across locations of the case study partners Large-Scale Agile Development Program

Noteworthy about the case is the high amount of Product Owners in each team in Figure4.1. Except for Team P, which functions as a platform team supporting the platform architecture, vision and managing the LSAD. The Product Owner of Team P is the Chief Product Owner, who is responsible for the whole product development. Similarly, both Scrum Master of Team P hold more product managing roles. While one focuses on the

correct use and promotion of the agile practices, comparable to an Agile Coach, the other is responsible for the technical development of the product. Nonetheless, the three members of Team P identify themselves as Product Owner and Scrum Master within that team, respectively. The program teams are geographically distributed over four locations, Figure4.2 shows the team composition with the respective locations.
As visualized in Figure4.2 every team is geographically distributed, which leads to challenges in communication and coordination as well as setting meeting times, fitting to all time zones. However, no team has members in all four locations, and the LSADP tries to have at least a few members per team working from the same location.
The LSAD studied at the case study partner supports the purchasing process of the whole organization. PILUM-Purchasing Integration, Lean and Unified Management- is the name of the product created by the LSADP at the case study partner. The program works on the

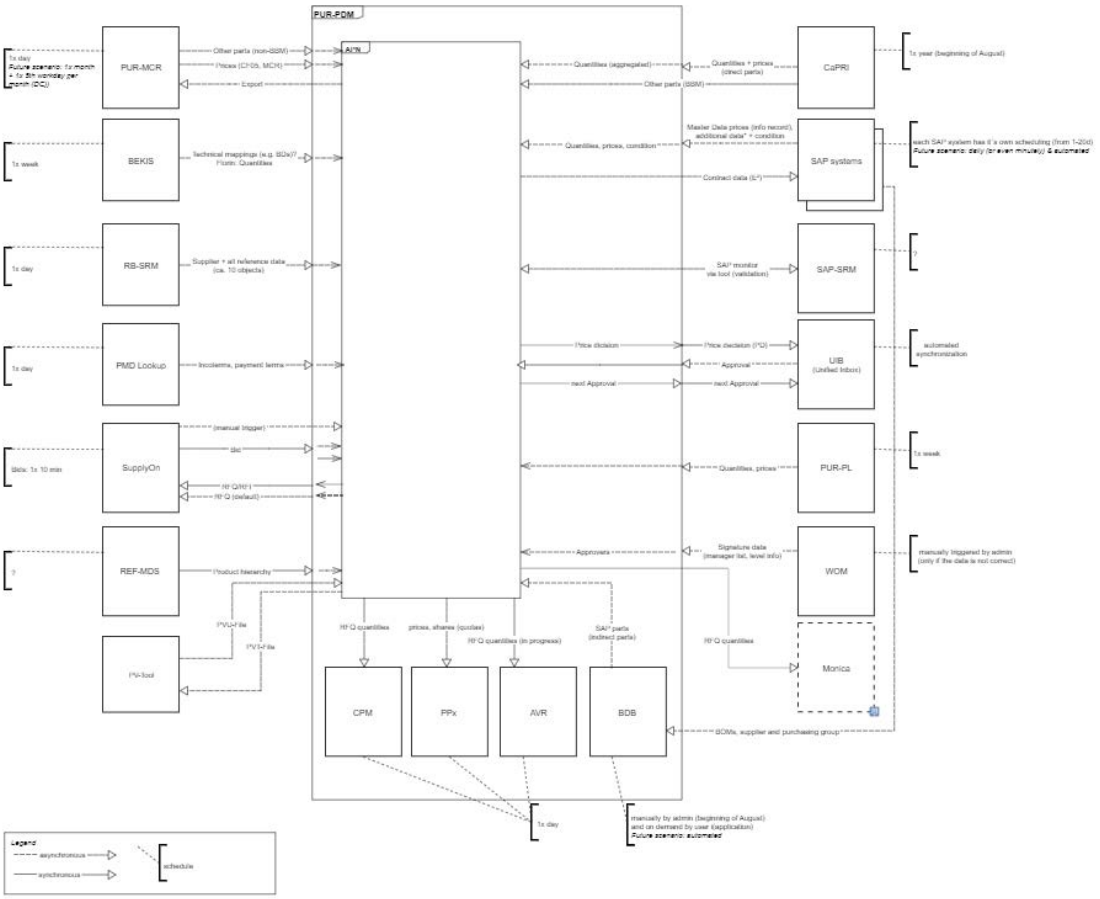


Figure 4.3.: APN-Module Architecture at Technology LLC.

PUR-PDM platform, which combines nineteen modules to bring to the user the PILUM product. An example for the complexity of one module is given by the module architecture diagram, in Figure4.3. The module referenced in Figure4.3 is responsible for delivering the information for annual price negotiations(APN). The module has roughly 1000 users worldwide and 4 internal and ten external interfaces. This one module is used by seven key user groups. As the platform combines seventeen modules of similar size and complexity, the scale of the platform becomes apparent. In total the platform has 17 external interfaces, 19 modules and nine key user groups. The LSAD the case study partner works in and detailed role, events and artifacts of the LSAD are presented in Section4.2.

4.1.2. General Information Interviews and Approach

For this thesis a total of 14 expert interviews were conducted, see Table4.2. Two types of interview styles were conducted for this thesis.

Interview-Partner	Duration	Interview-Type
Product Owner(PO-1)	1:06:21 h	semi-structured
Scrum Master(SM-1)	1:07:24 h	semi-structured
Product Owner(PO-2)	1:05:41 h	semi-structured
Scrum Master(SM-2)	1:23:33 h	structured
Product Owner(PO-3)	1:23:54 h	semi-structured
Product Owner(PO-4)	1:17:55 h	structured
Development Team(DEV-1)	1:12:48 h	semi-structured
Product Owner(PO-5)	1:13:40 h	semi-structured
Development Team(DEV-2)	1:02:52 h	semi-structured
Development Team(DEV-3)	0:50:42 h	semi-structured
Product Owner(PO-6)	1:13:52 h	semi-structured
Development Team(DEV-4)	1:16:28 h	semi-structured
Development Team(DEV-5)	0:54:52 h	semi-structured
Scrum Master(SM-3)	1:12:02 h	structured

Table 4.2.: Interviews conducted with Duration and type of questionnaire

One structured interview with the goal of gaining deeper knowledge of the LSADP, adapted from the work of Uludağ et al.[58] The other interview type was semi-structured with the goal to identify recurring concerns and good practices, adapted from the work of Uludağ et al.[55]. Both questionnaires were developed in the larger research project this thesis is a part of. The two questionnaires used for these interviews are included in the AppendicesA.1 andA.2. The Table4.2 shows the roles and interview types conducted. Additionally during the period of the thesis, observations of the LSADP, including the Kickoff of the program year 2020, were documented in notes. All observations and the nature of them are listed in Table4.3 For Section4.2 this thesis used three structured(see AppendixA.1) ex-

Observation-Event	Duration	Types of Documentation
Dailies	1:45 h	Notes
Newsflash	1:00 h	Notes
Sprint Review	7:30 h	Notes
Sprint Planning	0:45 h	Notes
Scrum of Scrums	0:30 h	Notes
Product Backlog Refinement	1:00 h	Notes
Kickoff 2020	14:00 h	Notes and Pictures

Table 4.3.: Observations from different team & program events

pert interviews along with the observations from Table4.3. For Section4.3 and Section4.4 the 11 semi-structured interviews were conducted, with questionnaire (see AppendixA.2) along with the observations regarding pattern candidates identified.

4.2. Large-Scale-Agile Development Program

This section describes the LSADP at the case study partner,(see Figure4.4). The data has been collected by conducting three expert interviews with PO-4, SM-2 and SM-3 using the Agile-Adoption questionnaire (see AppendixA.1) as well as observations from events and artifacts provided by the case study partner. We will begin with the introduction and implementation of the LSAD Section4.2.1. Then, we will introduce the Principles in Section4.2.2 of the LSADP,the Roles in Section4.2.3 with their responsibilities, the Artifacts in Section4.2.4 and Events in Section4.2.5 of the LSADP. Following, we will discuss how the topic of architecture (see Section4.2.6) is involved in the LSADP. At the end of each section will be a comparison between the foundation during the adoption process (LeSS and Spotify model) and the current LSADP of Technology LLC..

4.2.1. Agile Transformation

The Agile Transformation of Technology LLC.s' LSADP resulted in the process provided in Figure4.4 The Teams are represented by their flow line during the Sprint, the flows are organized from front to the back from Tam A to Team P, respectively. Team P's Sprint conception differs most from the other teams, this is partly due to the fact that as the platform team most of its team members are involved in several teams or are responsible for the overall success of the program. From the three structured expert interviews (see Table4.4) with SM-2, SM-3 and PO-4 the LSADP was described to function as follows. Technology LLC.s' LSADP started adopting scaled-agile practices in the beginning of 2017. As the program grew, more and more functionalities of what used to be separate systems were migrated into the new platform. With all these migrations the team grew until at last by the end of 2016 the decision was made to work in a LSADP and adopt the LeSS framework.

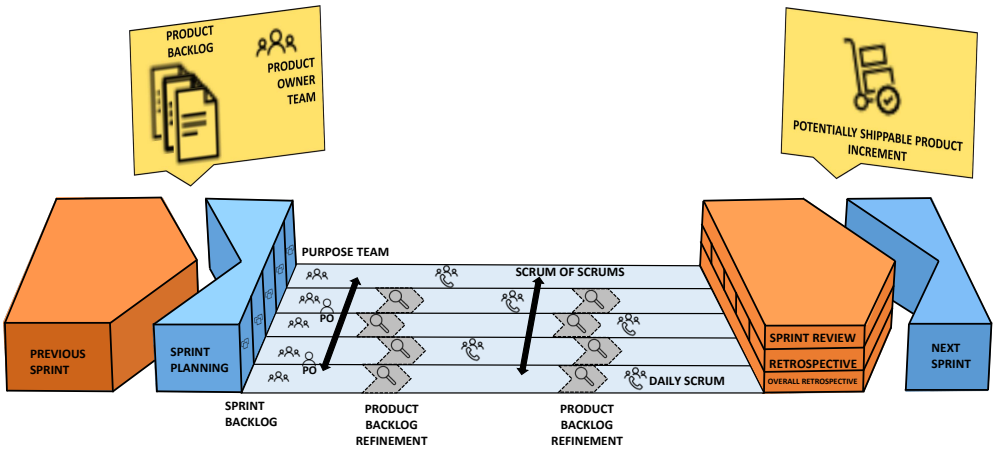


Figure 4.4.: Large-Scale Agile Development Program- Case Study Partner Technology LLC. . Adapted from the original framework by The LeSS Company B.V.[3]

Interview Partner	Experience
Product Owner (PO-4) - Chief Product Owner(CPO)	3-5 years
Scrum Master (SM-2) - Product Development Manager (PDM)	1-2 years
Scrum Master (SM-3) - Agile Coach (AC)	3-5 years

Table 4.4.: Interview partners for Agile Adoption questionnaire

'The introduction will never be completed, as we won't stop adopting our process.'. Scrum Master(SM-2)(2019).
Agile practices were not new to the case study partner as most people had experience with agile practices. However, the scaling of these agile practices was somewhat new to everybody. With the LSADP start at 2017 a transition phase was started in cooperation with an external coach, who helped them transform to LSAD. Additionally, a two-day workshop to introduce agile practices was held to start the transition phase.
'LeSSwas selected as it offers transparency, was the most Scrum like framework, yet, seemed easy to adopt.' Scrum Master (SM-3) (2019)
In addition to LeSS, the LSADP adopted some practices from the Spotify model as well as the Scrum of Scrums (SoS) from Scrum at Scale. Another add-on was the introduction of a Process Consultant, a role already in use in IT-projects at the case study partner, yet not relevant in LeSS.

'The modules , for purchasing, that we support require a highly domain-specific knowledge. We then introduced the Process Consultant as a single point of contact for development and product owners.' Scrum Master(SM-2) (2019)
Before the introduction of the PC role, the program had something they collectively call an 'Estimation Disaster'. The developers didn't have enough domain-knowledge to create accurate estimates. These inaccurate estimates and the limited domain-knowledge led to prolonged Product Backlog Refinements and over-commitment in practically every Sprint. During the Transformation phase, there were multiple challenges encountered by the LSADP. While they introduced scaled-agile to a team of roughly thirty employees, the program faced challenges in multiple areas:

- Correct application of agile practices - estimation disaster.
- Cultural difference and geographical distribution - four locations and three cultural backgrounds.
- Historical role distribution of the four locations.
- Self established teams- co-located teams formed, however, no focus on knowledge of the team members.
- Introducing LeSS without any adaptions.

In the beginning of the Agile Transformation, the LSADP had issues to correctly apply agile practices, they were stuck in a loop of pseudo-agility and did not focus on the values and principles of agile, but on the application of agile practices. The shift of responsibilities to the teams entailed that the program faced teams, who had issues to self-organize and deliver increments. By introducing the Process Consultants and creating common values regarding agile and focusing on the values and principles that are underlying in agile, the program was able to address these challenges. They acknowledged the domain-knowledge issue and led several discussions to understand their values, e.g. quality has a high value in the Warsaw location.
'Our colleagues from Poland value Quality and Sustainability more, while the German colleagues valued Delivering features higher.' Scrum Master(SM-2) (2019)
All three interview partners agree that enabling the teams, by forming Purpose Teams and introducing Process Consultants were the main factors to address most of their issues. As the Lesson Learned of their experience with agile transformation, they acknowledged that too much change at once leads to issues. They also highlighted that adapting a framework as is, probably will never solve anything. Their program became successful once they actively adapted their program to fit their needs.
'One major lesson learned is being confident enough to change a framework and create our own process, find agile practices that fit you, no matter where they origin. While simultaneously always questioning our own process and trying to improve it.' Scrum Master(SM-3) (2019)

Two interview partners mentioned to focus on incrementally changing a process and evaluating the changes constantly.
'Have a more organic growth and change, incrementally change and observe the effects the changes have on the process. Have the courage to dismiss practices, if they are not useful for you, even though they are very common. Product Owner (PO-4) (2019)
As it stands now the LSADP has established itself as a pretty secluded program in the context of Technology LLC.s' whole organization, see Figure4.5. Around 90% of the LSADP

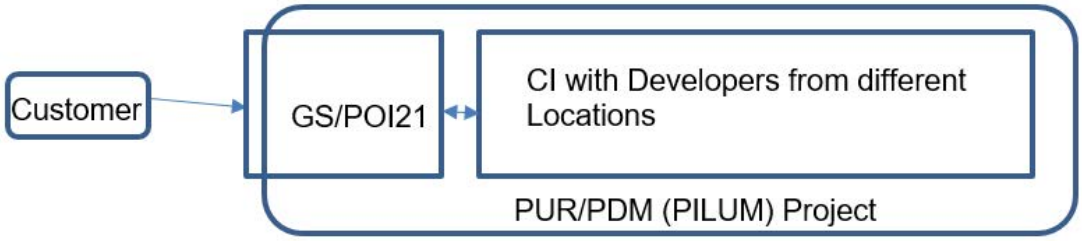


Figure 4.5.: Large-Scale Agile Development Program - Relationship of program in organizational context.

members have 100 % of their time allocated to the program, especially the development employees. The interview partners agreed on them applying LeSS and not LeSS Huge, because the Purpose Teams are not representative of a LeSS Huge adaptation. Further, they estimated to use about 70 % mainly Scrum practices adapted in LeSS and the other 30 % is a mix of LeSS practices and Spotify practices. The following stakeholders were identified by the interviewees:

- CI - Corporate IT - from our team members to the management.
- POI - IT project management and global services department - from our team members to the management.
- Customers - every key user department and department using our product.

In total, the stakeholder groups of the LSADP are identified similarly to Figure4.5 Overall, the LSADP went through two phases of agile transformation so far. Similar to the observed phases in the works of Fuchs and Hess[25] and Paasivaara et al.[39], the LSADP agile phases were initiated by a push to become more agile. During the end of this case study, there were first indicators of another agile transformation phase being initiated. The LSADP size was increased combining two LSADP into one LSADP.

4.2.2. Principles

Principles of LeSS have been adopted partially, yet the main focus of the LSADP lies on the agile values and Scrum principles.

'We have communicated and discussed what the values of Openness, Courage, Respect, Focus and Commitment mean to every single team in our 2018 Kickoff.'Scrum Master (SM-2) (2019)
With the Scrum principles at their core, SM-3 mentioned that since LeSS is a scaled version of Scrum, they also follow the LeSS principle of LeSSis Scrum. The LSADP elaborated on these values with statements on how they understand the concepts of :

- Openness - open for feedback, change, inform your team about scope changes/ problems, etc.
- Courage - admit failure, refuse overload, question things and discuss them, etc.
- Respect - listen, don't interrupt, respect work-life balance, etc.
- Focus - avoid topic jumps, focus on Sprint Goal, focus on customer needs, etc.
- Commitment - go the 'extra mile' where needed, take responsibility for your decisions, etc.

Internal coordination is implemented by CoP, SoS across the teams and all Scrum events for within the teams, every other communication is conducted in form of direct communication, as it is preferred by all teams. This can be summarized by the following statement: If you need someone's help, whether they are on your team or not, contact them directly. For topics affecting several teams, there are also the concept of Newsflash and the cross-team events of LeSS(Review, overall retrospective). Communication and Coordination to stakeholders is done either by involving a customer in the Sprint Review or through the periodic round-tables by the Product Owners. Regarding the LeSS principles, the LSADP focuses on Transparency, Customer-Centric and Continuous Improvement.
Transparency. Transparency is regarded as the most important principle in the LSADP as it enables most of the other principles. The teams are informed about anything managerial with a Newsflash, where they are allowed to ask questions and offer input on the topics discussed. The Sprint Goals and progresses of all teams are accessible for all program members in JIRA and Power BI- Microsoft Business Intelligence program, visualizing progress, detailed information on requirements and other Key Performance Indicators (KPIs).
Every member is allowed to visit other teams' events, to gain information about their process. Additionally, all CoP in the LSADP are open for anybody interested in the topic.
Customer-Centric. *'Customer is King' Product Owner (PO-4) (2019).* The main goal of adopting a LSAD was to improve customer satisfaction. Technology LLC.s' LSADP has nine key user departments, each with several key users and different needs for different products. The Commitment to make a customer-centric product is reflected in the Product Owner Team. Each Product Owner has several key users and modules he supports. They focus on creating personal relationships and regular coordination with each key user. With only one PO this would not be possible. The POs try to improve the modules they are responsible for and coordinate in doing so with their colleague POs, to minimize the effect on the

agile team.

Principles and Values	Technology LLC.s' LSADP	LeSS	Spotify
Scrum values(Large-Scale Scrum is Scrum)	x	x	x
Transparency	x	x	x
More withLeSS	x	x	
Systems Thinking		x	
Empirical Process Control		x	
Continuous Improvement Towards Perfection	x	x	x
Customer-Centric	x	x	x
Whole Product Focus		x	x
Queuing Theory		x	
Lean Thinking	x	x	

Table 4.5.: Comparison of the Principles and Values of the Large-Scale Agile Development Program at Technology LLC., LeSS and Spotify

Continuous Improvement. Using Retrospectives and living the values of courage and commitment helps enable continuous improvement. The teams are autonomous and can decide to change the Sprint process when they see fit and not when a manager or the platform team says so. Waste avoidance and correct application of the agile practices can be observed as well. During several observed meetings, we could hear a team member remind his colleagues to focus on the scope of the meeting. It is interesting to note that not always the same person, but everybody who thought the discussion or presentation is not fitted to the event, spoke out.
Additionally, we were able to be part of the Kickoff 2020 of the LSADP, an event where best practices in the program were presented to help all teams improve their process. Following the presentations each team then discussed on their own what they want to change about their process. In conclusion, the LSADP tries to constantly remove non-value adding events and artifacts and tailors its program towards what makes it more efficient and valuable to the customer.

Improvement Possibilities

Areas of improvement identified by our interview partners are Whole Product Focus and Commitment. Currently, there is a lack in commitment, not in the program itself, but in adhering to the use of artifacts and understanding the concept of ownership of a module. *'When we release an increment, how does the customer experience it' Scrum Master (SM-3) 82019).*
The introduction of Process Consultants slowly improves Systems Thinking, as there are more system thinkers involved in the process. On the other hand, the same role limits the knowledge share, as it takes over some decision making capabilities from the team and

the teams rely too much on the expertise instead of trying to solve an issue on their own. Table4.5 offers an overview of the Principles and Values theLSADP applies in comparison to the two frameworks it adapted.

4.2.3. Roles

Role	**Responsibilities**
Product Owner	Responsible for managing Product Backlog(clarifying functional requirements), customer communication and coordination, releasing increments for his module.
Scrum Master	Responsible for resolving impediments and bottlenecks and leading the Scrum events
Development Team	Responsible for Increment development, detailed Backlog item estimation, adhering to module Architecture and Sprint Review
Test Team	Responsible for manual testing
Process Consultant (*new*)	Responsible for initial backlog item estimation, module Architecture creation, aligning teams with Macro-Architecture and supporting Purpose Team through Knowledge Sharing and coaching
Chief Product Owner(CPO)	Responsible for functional product development facing customer, epic-plan fulfillment
Product Development Manager(PDM)	Responsible for personnel decisions and non-functional development of product
Agile Coach(AC)	Responsible for coaching and introducing correct agile practices and supporting SM in their role
Software-Architect	Responsible for introducing new programming concepts and leading the adoption –> Angularization,..
IT-Architect	Responsible for Macro-Architecture vision and concept, aligning teams in cooperation with PCs covering all non-functional requirements
Test Architect	Responsible for automated testing

Table 4.6.: Roles and Responsibilities according to expert interviews and observation notes

The roles and responsibilities are elaborated in Table4.6. An agile team, the LSADP

Roles and Team	Technology LLC.s' LSADP	LeSS	Spotify	Remarks
Feature Teams	x	x	x	The LSADP uses domain-feature teams called Purpose Teams
Product Owner	x	x	x	
Scrum Master	x	x		
Development Team	x	x	x	
Test Team	x	x	x	
Process Consultant	x			
Chief Product Owner	x	x	x	LeSS uses Chief Product Owners in LeSS Huge
Product Development Manager	x			
Agile Coach	x	x	x	Spotify has Agile Coaches as part of every team comparable to a Scrum Master
Software-Architect	x	x	x	Part of feature Team in LeSS. At Spotify Chapter Lead Development
IT-Architect	x	x	x	Part of Feature Team in LeSS. Spotify calls them System Owner(technical)
Test Architect	x	x	x	Part of Feature Team in LeSS. At Spotify Chapter Lead Test

Table 4.7.: Comparison of the Roles of the Large-Scale Agile Development Program at Technology LLC., LeSS and Spotify

calls them 'Purpose Team', usually consist of all roles required to develop features, including Product Owners and Process Consultants (see Figure4.1). Agile Coach, Chief Product Owner and Product Development Manager form the project management group. They discuss any program relevant issues, budget plans, need for developers and more. However, the project management group tries to make as few as necessary decisions affecting the autonomous teams. Improvement opportunities lie in the area of continued ownership. Currently, after releasing an Increment, the agile team loses the focus on ownership of that module. Therefore, limiting the commitment to improve non-functional requirements of the module.
'If the Testers were more involved in creating automated testing, it would immensely improve the workload of the developers and Test Architect. Scum Master (SM-3) (2019) The Test Team and manual testing is mainly done by an external organization, who has been working for a long time with the LSADP. Table4.7 offers an overview of the Roles the LSADP uses in comparison to the two frameworks it adapted.

4.2.4. Artifacts

Assigning Backlog items to the purpose teams is completed via labels in the ticketing system JIRA. Whenever a functional or non-functional requirement is created, the creator adds a module and epic label to the requirement. The module and epic label help to assign the requirement as all epics have been allocated to teams and the responsible PO then estimates and elicits the Backlog Item and brings it into the Product Backlog Refinement (PBR) and when finalized into the sprint.
'The Backlog is managed by the POs, they check with the epic plan when an Epic is supposed to be implemented and then bring the items respectively in PBRs and assign them to the Sprint.' Product Owner (PO-4) (2019)
Through the creation of the epic plan and the teams being domain-specific experts, the assignment of backlog items has been managed. In a final step the PO assigns an item to a specific Sprint. The other artifacts in use at the LSADP are summarized in Table4.8. The interview partners highlighted that there is no confusion as to who is responsible for the artifacts and that they are overall happy with how the artifacts are used and the purpose they serve.
'We implemented 65 Epics in 2018, 8 Epics per PO. Hence, there is only little time to try different solutions and create Minimum Viable Products to test. ' Scrum Master (SM-2) (2019)
SM-2 sees a need to introduce Minimum Viable Product (MVP) in the future. MVPs would need more information in the creation process of Epics. Currently, the customer creates very abstract epics in the planning of the next year and the team doesn't have time to elicit further or create solution concepts via MVPs as there are not sufficient information or resources available. Hence, there is a need to make the epics more accurate in the beginning and add more manpower to allow an easier process. Table4.9 offers an overview of the Artifacts the LSADP uses in comparison to the two frameworks it adapted.

Artifact	Content	Responsible for Artifact
Product Backlog	all requirements functional and non-functional	Product Owner Team
DoR and DoD	what is required for a backlog item to be ready for implementation(DoR), ready for increment release(DoD)	Scrum Master for maintaining and Agile Team for creation
Sprint Goal	Main goal to be reached through Sprint, what is the tagline of this Sprint	Team in Sprint Planning
Sprint Backlog	Items assigned from the Backlog, to be implemented during Sprint	Product Owners
Acceptance Criteria	what needs to be done for a specific item, so it complies with the functional requirement	Product Owner
Epics	larger requirement creating business value for customer	PO
Increment	functional requirement collection, which delivers a part of an Epic	Purpose Team

Table 4.8.: Artifacts of the Large-Scale Agile Development Program according to expert interviews and observation notes

Artifacts	Technology LLC.s' LSADP	LeSS	Spotify	Remarks
Product Backlog	x	x	x	
DoR and DoD	x	x		LeSS uses DoD
Sprint Goal	x	x	x	
Sprint Backlog	x	x	x	
Acceptance Criteria	x			
Epics	x			
Increment	x	x	x	

Table 4.9.: Comparison of the Artifacts of the Large-Scale Agile Development Program at Technology LLC., LeSS and Spotify

4.2.5. Events and Process

The LSADP process depicted in Figure4.4 consist of all events directly associated with the Sprint process of Technology LLC.. The CoP event is not included in Figure4.4 as it made the process unclear. CoP's and their events are further discussed in Section4.2.6.

Sprint
The goal of the Sprint is to implement the Sprint Backlog and create a product Increment. All purpose teams work in a synchronized Sprint. The Sprint occurs over a three week period.

Sprint Planning
The goal of Sprint Planning is to elicit the Sprint Backlog in each Team, with all team members present. It occurs once a Sprint at the beginning. The whole team is present and the head PO presents the Sprint Backlog items, the team votes to accept or reject these items.

Daily
The goal of a Daily is getting an overview of each team members progress. All team members are present during the Daily, which takes 15 minutes. Team members present yesterdays and today's work and whether they have any impediments.

Scrum of Scrums The goal is an overview of all teams and to help coordinate work between the teams. All Scrum Masters and eventually representatives of a team with an impediment are present. SoS occurs three times a week, after the dailies. The Participants talk about the progress in teams and whether there are any impediments in a team.

Process Consultant Meeting ***new***
The goal is to align Architecture and create new concepts. The PCs and Architects are the participants. It occurs once a week for one hour. The Architect presents new concepts or the progress of current adoption, followed by a discussion on how to move on and whether there are any limitations on a module side.

Community of Practice
The goal of CoP is Knowledge Sharing and generating common practices for a specific role or topic. Anyone interested in topics can participate. CoP occur once a sprint for one hour and are lead by a Moderator. Discussions are in accordance with the agenda(Types ofCoP at Technology LLC. :TestCoP, DevelopmentCoP, UI-CoP,...)

Product Backlog Refinement
The goal is to clarify Backlog Items, which are to be implemented in future sprints and estimating the items. All members of purpose team participate. Once a week during Sprint or once a Sprint (Team decides). PO goes through a prepared list of items and the team discusses the items and how to solve the item. Afterwards the team estimates the required work.

Newsflash
The goal is to present managerial information to all teams. Therefore, all purpose team members should be present, however it is optional. Occurs once a month. Product Development Manager and Chief Product Owner present any managerial topic influencing all purpose teams,making transparent the work of the project management group.

Team Retrospective
The goal is to identify what went well during the sprint and what needs to improve, challenging the process and possibly adapting the process. All team members of purpose team are present. Retrospectives occur after every second sprint, again the teams can decide to do them more or less. The SM leads the discussion to create a better process in the future. It should not take longer than three hours.
Overall Retrospective
The goal is a program-level Retrospective. The Agile Coach wants to figure out what needs to be addressed on a program level. The Agile Coach and representatives of teams (max. 2 per team) are participating. The Overall Retrospective occurs twice a year. The Agile Coach leads discussion on possible change opportunities in the applied agile practices.
Sprint Review
The goal is to present work done during the Sprint of each team. All purpose teams and all team members are participating. The Review occurs after the Sprint. The Teams present their work.
'I'd like to hold the Overall Retrospective with more participants, right now we only have the

Events	**Technology LLC.s' LSADP**	**LeSS**	**Spotify**	**Remarks**
Sprint	x	x	x	
Sprint Planning 1		x		
Sprint Planning 2	x	x	x	
Daily	x	x	x	
Scrum-of-Scrums	x	x		
Process Consultant Meeting	x			
Community of Practice	x	x	x	Spotify uses Guilds, which are comparable to CoP
Product Backlog Refinement	x	x	x	
Newsflash	x			
Team Retrospective	x	x	x	
Overall Retrospective	x	x		Only very limited use at LSADP of Technology LLC.
Sprint Review	x	x	x	

Table 4.10.: Comparison of the Events of the Large-Scale Agile Development Program at Technology LLC., LeSS and Spotify

Scrum Masters and there is no detailed insight into other roles' perception of the process.' Scrum

Master (SM-3) (2019). For the LSADP, the PBR is a really important event as it allows detailed discussions and estimations. Due to the many Epics, the PBR helps clarify the actual Business Value, which an Epic aims to achieve. The interview partners plan to introduce an event called Epic Preview to help clarify epics to the teams and create a little kickoff for the epics. Overall, the major changes between the LSADP and the the two foundations Technology LLC. chose for adopting agile, can be found in the events and process. The LSADP removed Sprint Planning 1 and used the concept of Sprint Planning 2, as a single Sprint Planning. The LSADP Sprint Planning is equal to Sprint Planning 2, they identified that Sprint Planning 2 delivered the outcome they needed and the additional Sprint Planning 1 was inefficient.
Additionally the LSADP uses PBR and SoS meetings more proficiently than is promoted by LeSS. The final differences are they had to introduce the Process Consultant Meetings to address coordination efforts on architecture level and that they use the Newsflash to inform all teams transparently about managerial decision processes. Table4.9 offers an overview of the Events the LSADP uses in comparison to the two frameworks it adapted.

4.2.6. Architecture

Architecture topics are developed and envisioned by Team P with the IT-Architect taking charge. Team P (see Table4.1) as a platform team combines several architecture roles. The two Software Architects are responsible for pushing topics like Angularization or other development oriented topics. The IT-Architect creates a vision for the Macro-Architecture and develops and communicates it with the Process Consultants. Aligning the modules to the macro-architecture is then completed by the teams with the process consultants. The platform team creates the pipeline to enable the purpose teams in releasing their increments, similar to the example shown in Spotify (see Figure2.7). Additionally, the Test Architect leads the CoP-Test and enables the developers in creating automated tests. Architecture development appears to be integrated in the overall process, however, the vision and most of the development associated with architecture is still developed by the IT-Architect. *'90 % of architecture related development was handled by the IT-Architect.'Scrum Master (SM-2) (2019)*
The introduction of the Process Consultant Meeting aims at addressing that issue. In the Process Consultant Meeting architectural concepts are created for the modules as well as the alignment and coordination of macro-architecture topics is discussed. The Process Consultants then coordinate in the teams the architecture topics. Another way to communicate architecture and communicate already available solutions are the CoPs at the LSADP, see Figure4.6. Agile Master, (compare Figure4.6) is the internal name for Scrum Master and the CoP are held in a larger context. However, the other platform will remain unnamed. The CoPs address knowledge sharing and practice alignment and help members to solve their impediments. The overall construct presented in Figure4.6 includes practices adapted from the Spotify model (see Figure2.6). For documenting the architecture of the modules (see Figure4.3) and the platform as a whole, the LSADP uses the

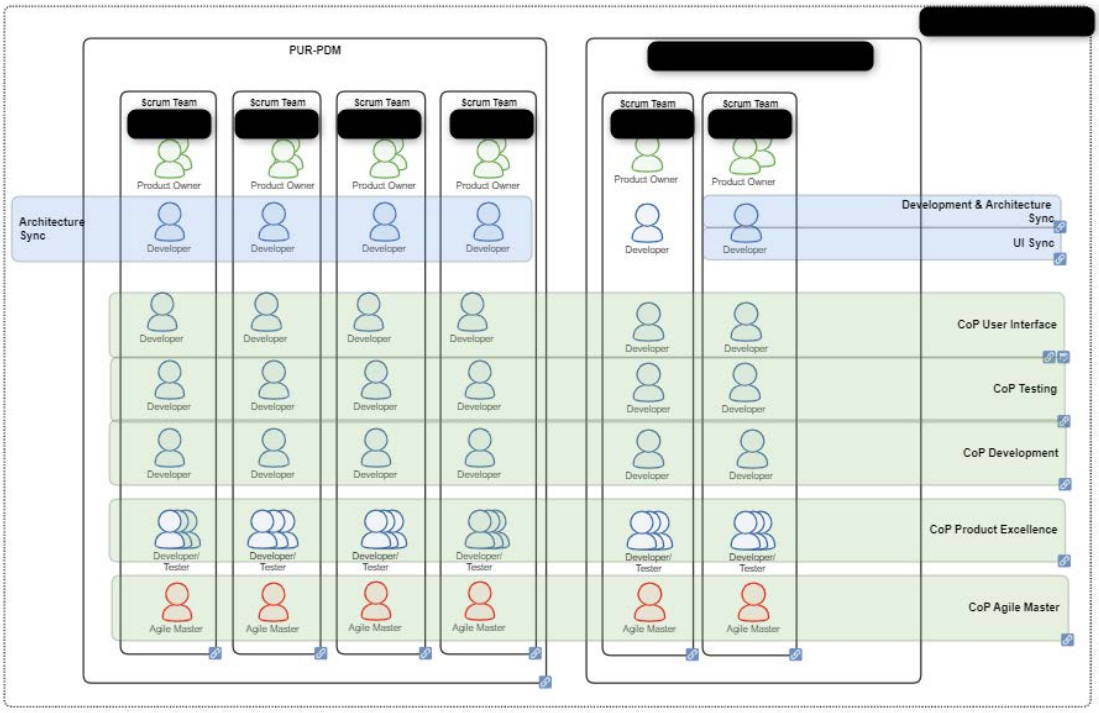


Figure 4.6.: Communities of Practice Structure, figure from Technology LLC.

arc42-template, created by Starke and Hrsuchka[52], to document the platform's architecture. The steps adopted by the LSADP from arc42[51, 52] are:

1. Introduction and goals (describe requirements, quality goals and stakeholders)
2. Constraints (any constraints)
3. Context and Scope (specify external interfaces)
4. Solution Strategy (Summarize decisions and solution strategies shaping architecture)
5. Building Block View (static decomposition of system)
6. Runtime View (behavior of building blocks in scenarios)
7. Deployment View (mapping building blocks to infrastructure elements)
8. Crosscutting Concepts (solution approaches view cross-cutting the system)
9. Architectural Decisions (decisions including rational)

10. Quality Requirements (high-level overview of quality goals from 1.)
11. Risks and Technical Debt (known risks and technical debt)
12. Glossary (term definitions)

All interview partners described their process of dealing with technical debt worthy of improvement. Currently, they resolve technical debts in a technical Sprint, which takes place at the end of each year, from the middle of December to the middle of January of the next year.
'We consciously collect our technical debt over the year. Trying to avoid as much as possible through test automation. In the technical sprint we then prioritize and work on our collected technical debt.' Scrum Master (SM-3)(2019)
The Goal for the next year is to take care of some technical debt during the year, improving the Quality of the platform continuously. The overall goal for architecture related topics is to make the progress more visual, to create a clear vision and effectively communicating it to all program members. This will foster cooperation between architects and the teams even further.

4.2.7. Summary Large-Scaled-Agile Development Program at Technology LLC.

In total, the LSADP was satisfied with LeSS, due to the ease of implementation and adaption. However, they don't think that the framework offers enough practices to address all their needs, which is why they combined it with the CoP culture of the Spotify model and introduced several changes. The complete differences between the frameworks and the current LSADP are summarized in Tables4.5,4.7, 4.9 and 4.10. The interview partners and the LSADP are overall satisfied with their adoption of LeSS and Spotify, nonetheless, they are focused on improving their program by identifying best practices to apply an integration in their program.
'What really makes me worry are external processes like budget planning, because they paralyze our functioning process. Here I'm missing the connection between theory and real-life. Product Manager (PO-4)(2019) To summarize the main concern, PO-4 states from the LSADP perspective all interfaces to non-agile processes(budgeting, traditional teams, external interfaces with traditional teams) are difficult to manage correctly.

4.3. Identification of Recurring Concerns

This section consists of the identified Recurring Concerns for the three stakeholder groups interviewed. These have been identified and documented using the LSADPL introduced by Uludağ et al.[55].

4.3.1. Interviews General Information

The interview partners, their role and experience in that role at the LSADP are summarized in Table4.11.

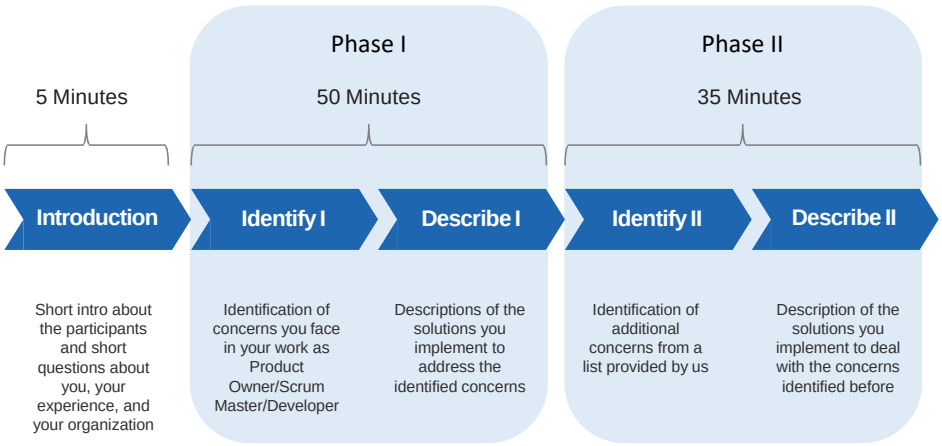


Figure 4.7.: Interview Process for Identifying Recurring Concerns, Good Practices and Bad Practices by Uludağ et al.[55]

The semi-structured interviews were conducted following the approach created by Uludağ et al.[55], see Figure4.7. Each of the 11 interviews using the questionnaire provided in A.2 consisted of two similar phases in Figure4.7. During the first phase, the interviewee

Interview-Partner	Experience
PO-1	3-6 years
SM-1	more than 6 years
PO-2	1-3 years
PO-3	1-3 years
DEV-1	3-6 years
PO-5	less than 1 year
DEV-2	3-6 years
DEV-3	1-3 years
PO-6	1-3 years
DEV-4	1-3 years
DEV-5	1-3 years

Table 4.11.: Interviews for Identifying Recurring Concerns, Good and Bad Practices

explained his role and concerns he faces working in the LSADP. Once a recurring concern was identified, the interviewee was asked to elaborate to make sure it was a new recurring concern he described and not an already existing recurring concern identified by Uludağ et al.[57].
In the second phase, the interviewer presented the interviewee with a list of already identified concerns in the research project by Uludağ[57] for the stakeholder group. In both of those phases, when identifying a concern, the interviewee was asked whether he had a solution for the respective concern.
Solutions mentioned were documented following the LSADPL by Uludağ et al.[55] into the categories of practices(Good or Bad Practices). The full list of newly identified concerns, good and bad practices are listed in the appendicesB.1 and C. In this section we will only introduce a few examples of concerns for each stakeholder group(PO,SM and DEV) interviewed. Additionally, the practices identified will be introduced in Section4.4 and the Mapping for each stakeholder of concerns and pattern candidates will be provided in Section4.4.15.

4.3.2. Recurring Concerns

This part is separated into the three interviewed stakeholder groups. DEV, PO and SM. For each stakeholder group the newly identified recurring concerns through the interviews are introduced. Additionally, an overview of all identified recurring concerns of a stakeholder group is provided. The full list of all recurring concerns- new and already existing-identified is provided in the appendixB.

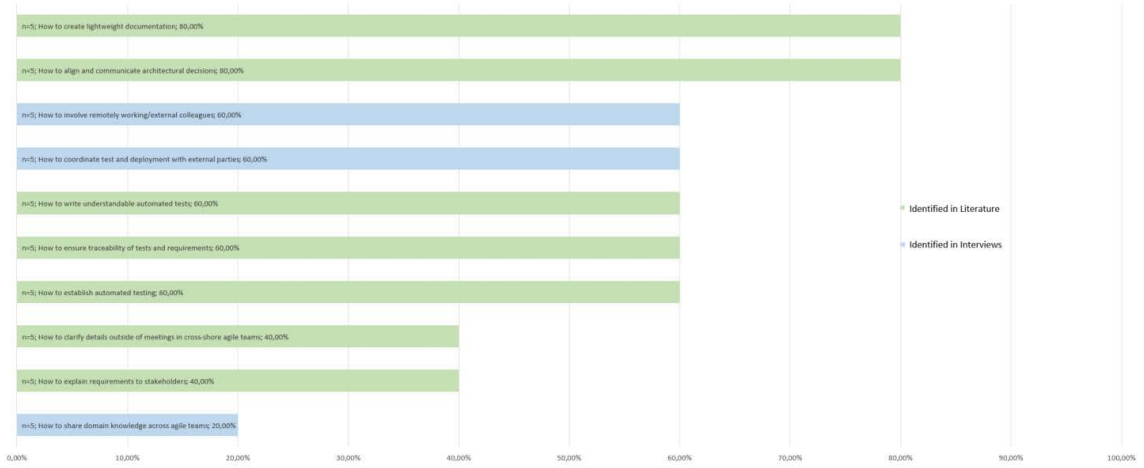


Figure 4.8.: Identified recurring Concerns of Development Team at Technology LLC.

Development Team

Recurring Concerns of DEV have not been studied a lot in the larger research project this thesis is a part of. With LSAD and Scrum trying to focus the DEV solely on development issues, the outcome of the interviews with DEV deliver a measure on how successful this approach is. However, as described in Section4.1 the LSADP works with external interfaces, geographical distribution and complex,legacy system. DEV have to navigate those challenges, and during the interviews identified the following **new** recurring concerns. In total, three new recurring concerns were identified by the interview partners DEV-1,DEV-2 and DEV-5.

- **C-85** *How to share domain knowledge across agile teams?*DEV-1 identified, that he as a developer has issues to gathering knowledge from other agile teams, when not having to deal with a major bug or system failure. The sharing of domain knowledge only occurs in teams of that domain and developers interested in the domain knowledge have a higher boundary to receive that knowledge. DEV-1 himself had issues sharing his knowledge with other agile teams in an efficient manner.

- **C-86** *How to involve remotely working and external colleagues?*dev-2 mentioned that the remotely working and the external colleagues had issues connecting to the videochats without technical issues. In case of remotely and external working colleagues this was limited to when they were not working from the LSADP offices. This is a concern for the Scrum Masters.

- **C-87** *How to clarify details outside of meetings in cross-shore agile teams?* DEV-5 had the recurring concern to get a hold of team members not co-located, which can especially painful when trying to gather some explanation to a question, which in turn makes

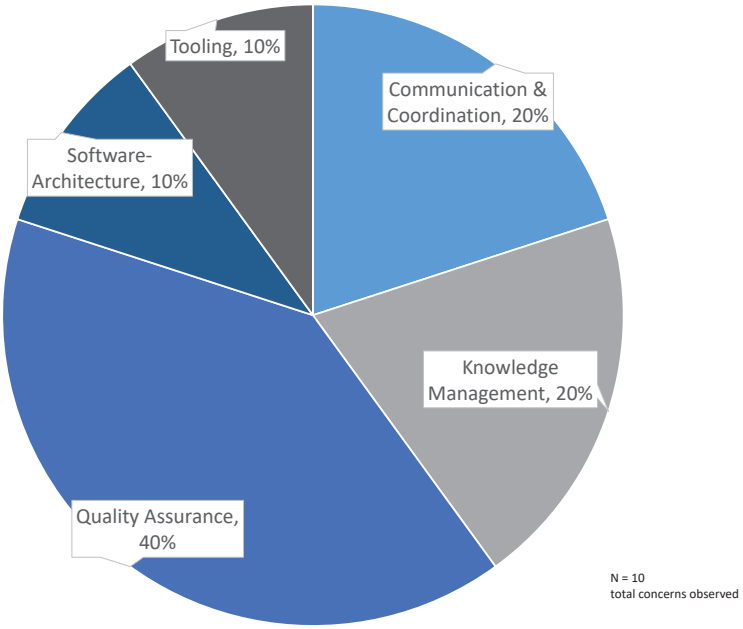


Figure 4.9.: Occurrence Concern Categories Development Team

the developer have to change tasks until he receives the answer. While communication is usually direct, sometimes due to the time difference, developers still have to wait a few hours to clarify an issue.

These three new concerns were checked with the identified concerns of Uludağ[57], to see whether they were different to already observed ones . While these concerns were identified there were a lot of DEV who identified **C-19** *How to deal with internal silos?* as a concern. However, due to the fact it represents a concern for the SM in the literature, this thesis omitted it in regard to the DEV. Newly identified concerns, associative to another stakeholder's challenge, were kept at the stakeholder group that identified the concern in this thesis. The combination of new and existing identified concerns was then presented to all DEV-interviewees and the final result of occurrence of each concern is presented in Figure4.8 In literature identified concerns and the new concerns, in combination with observed percentage and number of interviewees asked are summarized in Figure4.8. Main

assumptions developed from analysis of the identified recurring concerns of the stakeholder group DEV are:

- **C-26** *How to align and communicate architectural decisions?* was identified by 80% and has been included to the DEV as the LSADP aims at involving the DEV in the architectural decision making process, by making the Process Consultant a 50% member of the DEV.
- **C-37** *How to create lightweight documentation* was identified by 80 % of the DEV team and is categorized as a Knowledge Management concern. Further highlighting the concern in the category Knowledge Management at Technology LLC..
- **C-36**, **C-53**, **C-68** and **C-76** are all concerns regarding Quality Assurance, this identifies Quality Assurance as a major concern category for the DEV of LSADP.

The overall occurrence of concern categories is depicted in Figure4.9. 40% of the concerns identified by the DEV-interview partners were allocated to the Category of Quality Assurance, while 20% each were from Communication & Coordination and Knowledge Management and 10% each were from Tooling and Software-Architecture.
In total, the interviews with the stakeholder group DEV identified ten recurring concerns, of which three were newly identified and seven existing from Uludağ et al.[57].

Product Owner

There were already some case studies regarding recurring concerns of PO, however, so far they were not of multiple POs as a direct part of an agile team. For the Product Owners there were a total of 5 interviews with the A.2. In these five interviews the following new recurring concerns were identified.

- **C-79** *How to balance amount and quality of delivered requirements?* Even though as a Product Owner, one's main goal is to further deliver business value to customers, the balance of quality and delivery has to be kept. However, how do you make customers aware of the benefits of delivering less, but of higher quality?
- **C-80** *How to manage overarching backlog item prioritization with multiple product owners?* With a Scaled-Agile development Program consisting of at least Product Owners per team, the POs have to coordinate with each other which backlog items make the cut to be added to the Sprint Backlog. Prioritization is difficult and conflict will ensue, if there is no clear method for dealing with this concern.
- **C-81** *How to understand all interfaces and dependencies of the system?* Interviewee PO-5 described the issue of understanding all dependencies between the large amount of modules in the system. When creating user stories and requirements, it becomes difficult to then make sure the whole system is considered with a limited view of the system. This is a concern for the Enterprise Architect.
- **C-82** *How to support an On-Boarding approach for different stakeholder groups?* This is a concern for Scrum Masters, with a growing project and PO being a part of the agile team, the On-Boarding material has to be reflecting the different viewpoints of all stakeholders accurately. Otherwise, On-boarding is an issue.
- **C-83** *How to manage requirement development for multiple teams?* In case of PO-6, who is part of multiple agile teams at once, it is an issue to manage development at these different teams and continue having regular check-ins with customers. Keeping track of all developments becomes more difficult, as multiple modules can be concerned and interaction with multiple external partners might be required.

While these concerns were identified there were a lot of PO who identified **C-19** *How to deal with internal silos?* as a concern. Due to the fact that it represents a concern for the SM in the literature, this thesis omitted it in regard to the PO. These new concerns were cross-checked with the existing concerns identified by Uludağ et al.[57] to see whether any of them already exist in some manner. However, none of the newly identified concerns were mentioned before in literature, as the LSADP introduces multiple POs as part of one team, the concerns are so far unique to Technology LLC.. New concerns and identified existing concerns were then combined and presented to all POs interviewed, including PO-4, to get a bigger picture of how present the concerns are at the LSADP. Out of the newly identified recurring concerns, two (**C-81** and **C-82**)are categorized as Knowledge Management

concerns, which even though they are associated and solved by other stakeholder groups, were kept at the PO for this thesis as they identified the concerns. The total occurrences of all identified recurring concerns is summarized in Figure4.10. Main assumptions de-

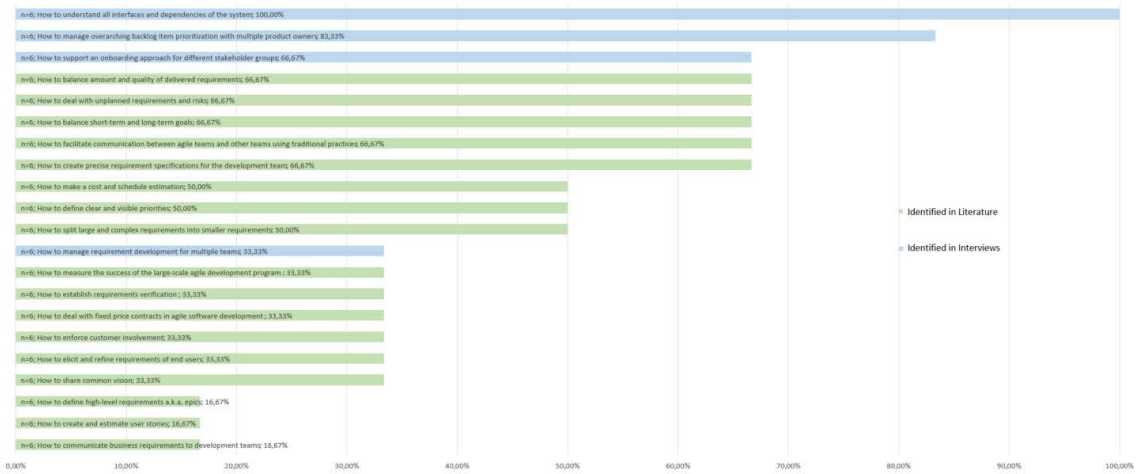


Figure 4.10.: Identified recurring Concerns of Product Owners at Technology LLC.

veloped from analysis of the identified recurring concerns of the stakeholder group PO are:

- **C-81** *How to understand all interfaces and dependencies of the system?* is a concern of all PO interviewed. Through the domain-specific encapsulation of the teams and the assignment to some modules per PO, the overall system is incomprehensible for the POs.
- **C-80** *How to manage overarching backlog item prioritization with multiple product owners?* Was identified by 83 % of the POs to be a concern of theirs. Unique to the LSADP of Technology LLC. so far, this concern stems from having multiple POs in one team. The concern is categorized as Communication & Coordination and can limit the effective use of the Purpose Teams.
- **C-10**, **C-15**, **C-18**, **C-22**, **C-28**, **C-35**, **C-60**, **C-69** and **C-70** are all concerns regarding Requirements Engineering, identifying Requirements Engineering as a major concern category for the PO of LSADP.

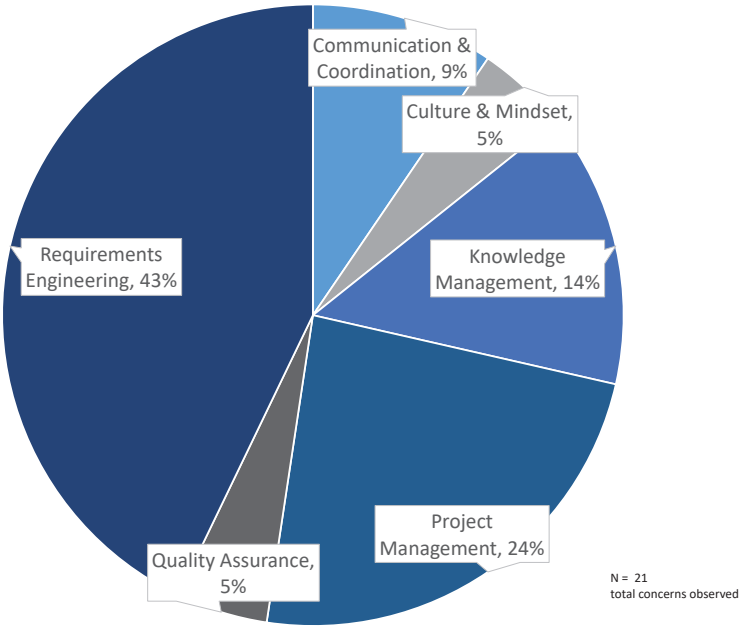


Figure 4.11.: Occurrence Concern Categories Product Owners

The overall occurrence of concern categories is depicted in Figure4.11. With a total of 43% of the identified 21 concerns for POs categorized as Requirements Engineering, 24% Project Management, 14% Knowledge Management, 9% Communication & Coordination and 5% each Culture & Mindset and Quality Assurance. In total, of the 21 observed concerns for the PO, 5 were newly identified by the interview partners and the remaining 16 were existing concerns identified by Uludağ et al.[57].

Scrum Master

The focus on SM has been done already in this larger research project, however, with Team P's Agile Coach and Product Development Manager identifying as SM in the LSADP, and the SM role being a 50% or less role for practically all SM it was interesting to see how that would influence the concerns observed. Due to the special situation, the SM actually identified several concerns of the already existing identified concerns by Uludağ et al.[57]. In total, the SM interviewed identified 35 concerns of which only one was newly identified through the interview process. The full list of 35 concerns was then cross-checked with all SM being a part of this case study interview process and the result is summarized in Figure4.12. The newly identified concern is **C-84** *How to involve all team members in solution generation?* and was identified by Scrum Master SM-1 within his team. The introduction of Process Consultants, while useful for many reasons, limited the engagement of other developers. When only one developer actively works on solution generation and all other team members accept his solution, the solutions are limited to one mind, instead of using the knowledge of a whole team.
C-84 when taking a closer look, differs from **C-67** *How to encourage development teams to talk about tasks and impediments?* and **C-74** *How to empower agile teams to make decisions?* both identified by Uludağ et al.[57]. While the already discovered concerns address the communication and decision making capabilities of agile teams, the newly introduced role of Process Consultant adds a need to specify the solution generation process of the agile team. Hence, **C-84** is not already represented and added to the list. The total occurrences of all identified recurring concern for the SM are summarized in Figure4.12 Main assumptions developed from analysis of the identified recurring concerns of the stakeholder group SM are:

- 12 out of 35 concerns are categorized as Culture & Mindset, the SM of the LSADP face most of their concerns in the area of application of agile practices.
- Geographical Distribution delivers three concerns all SM interviewed face, with the teams being distributed to at least two locations for all teams, see Figure4.2.
- **C-84** *How to involve all team members in solution generation?* as a newly identified concern is directly connected to the introduction of Process Consultants at LSADP.

The overall occurrence of concern categories is depicted in Figure4.13. With a total of 35 concerns identified, 34% are of the category Culture & Mindset, 17% Geographical Distribution, 14% Project Management, 11% Communication & Coordination, 9% each Knowledge Management and Methodology and 6% Tooling. During the interview process with the SM a list of what used to be eleven concerns, directly connected with the stakeholder group SM as identified by Uludağ et al.[57] grew to become 35 concerns observed. The additional 24 concerns identified were all due to the fact that SM in the LSADP hold multiple roles and the introduction of the purpose team, Process Consultant and the use of several PO actually increased the workload of the SM.

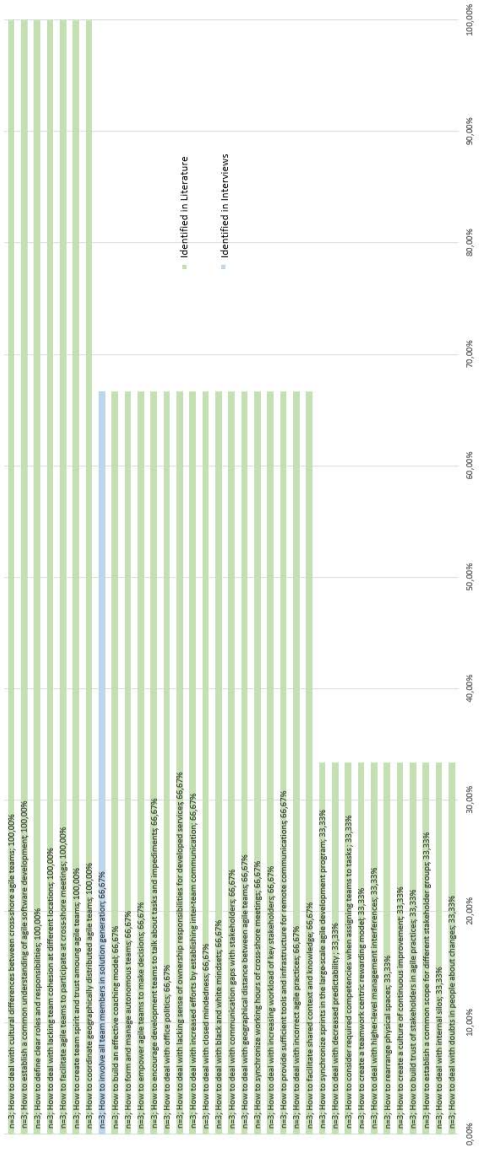


Figure 4.12.: Identified recurring Concerns of Scrum Master at Technology LLC.

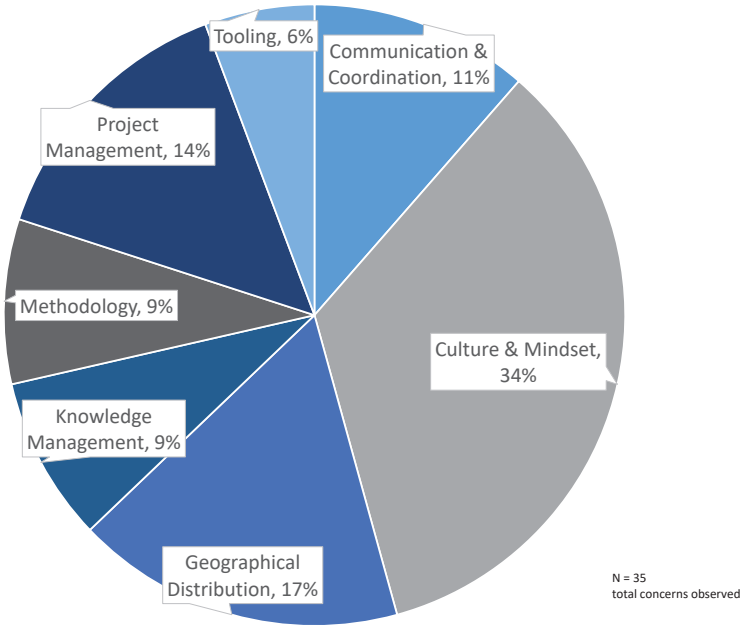


Figure 4.13.: Occurrence Concern Categories Scrum Master

Fittingly in the interview of SM-2, exactly that situation was addressed regarding the responsibilities of the agile roles. *'...apart from that, I think we need to invest more into the development of our Scrum Masters next year.' Scrum Master (SM-2) (2019)*

Overview

This section offers an overall overview of all occurrences of the 66 concerns (see Figure4.15) identified and the overall category distribution at Technology LLC., see Figure4.14 . The

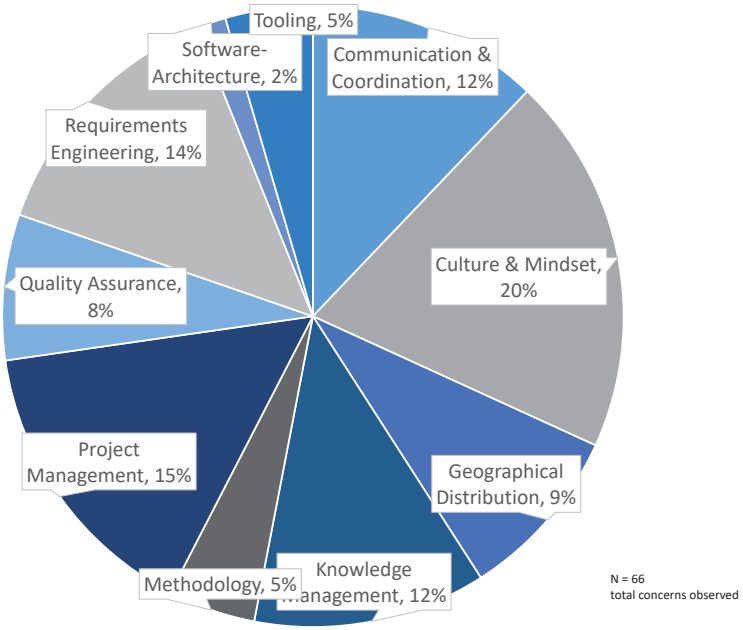


Figure 4.14.: Occurrence Concern Categories All

concerns with the most occurrences of each stakeholder group are of the category:

- DEV - Knowledge Management **C-37** and Software-Architecture **C-26**
- PO - Knowledge Management **C-81**
- SM - Culture & Mindset **C-24**, Geographical Distribution **C-03**, **C-29**, **C-32** and **C-61**, Methodology **C-59** and Project Management **C-56**

The category with the most concerns for each stakeholder group is:

- DEV - Quality Assurance, see Figure4.9

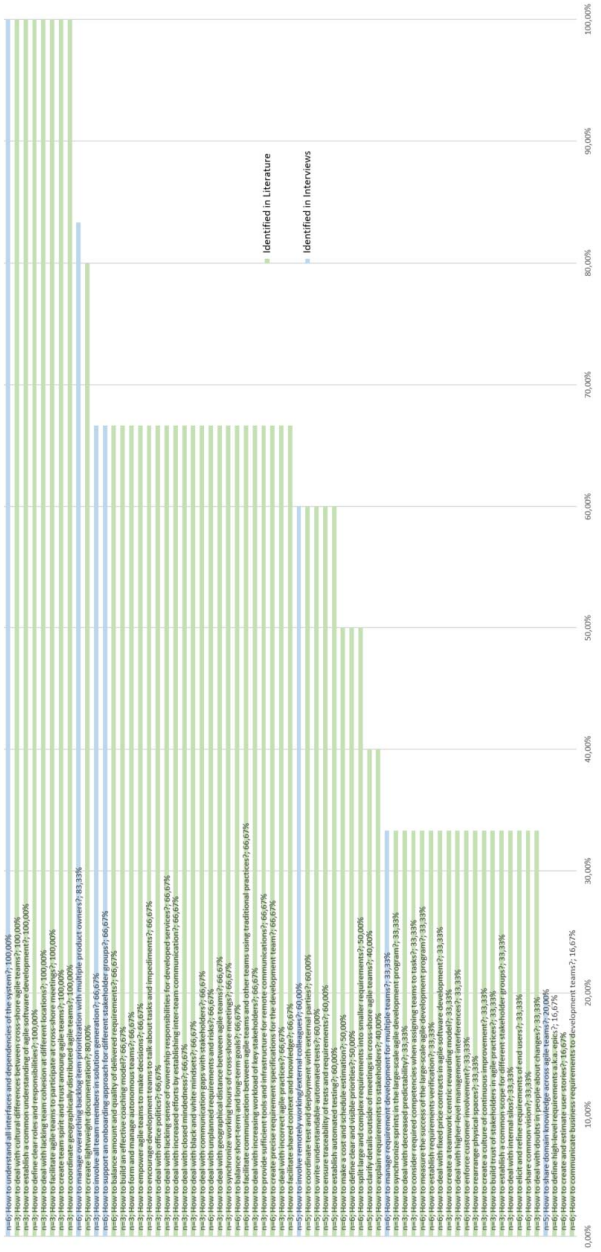


Figure 4.15.: Identified recurring Concerns of stakeholder groups at Technology LLC.

- PO - Requirements Engineering, see Figure4.11
- SM - Culture & Mindset, see Figure4.13

Additionally, the area of Knowledge Management and especially the concern **C-19** *How to deal with internal silos?*, identified by Uludağ et al.[57], was in some kind mentioned by all stakeholder groups. In total, nine new concerns were identified, three of the category Knowledge Management, two Communication & Coordination and one each for Culture & Mindset, Project Management, Tooling and Quality Assurance.

4.4. Documenting Good Practices and Bad Practices

This section consists of the documented Good and Bad Practices. The good and bad practices have been documented with the LSADPL introduced by Uludağ et al.[55]. The interview partners and their role and experience in that role at the LSADP are summarized in Table4.11.
We will introduce the two identified Patterns at Technology LLC. and examples of each Pattern Candidate Type documented out of the 50 total good practices identified(see Figure4.16). During the interview process a total of 50 pattern candidates were identified and documented.
Since this is a single organization case study, a high amount of pattern candidates and a low number of patterns results. Therefore, if a the pattern candidate was not exactly the same as already documented in the Pattern Catalog[61], this thesis documented a Good or Bad Practice instead, which creates a Pattern Candidate. The Pattern Candidate Types are distributed as follows:

- 11 Coordination Pattern Candidates.
- 18 Methodology Pattern Candidates.
- 7 Viewpoint-Pattern Candidates.
- 11 Anti-Pattern Candidates.
- and 3 Principle Candidates.

Additionally, two already observed patterns were identified at Technology LLC. **CO-1** *Communities of Practice* and **A-1** *Don't use Frameworks as a Recipe*.

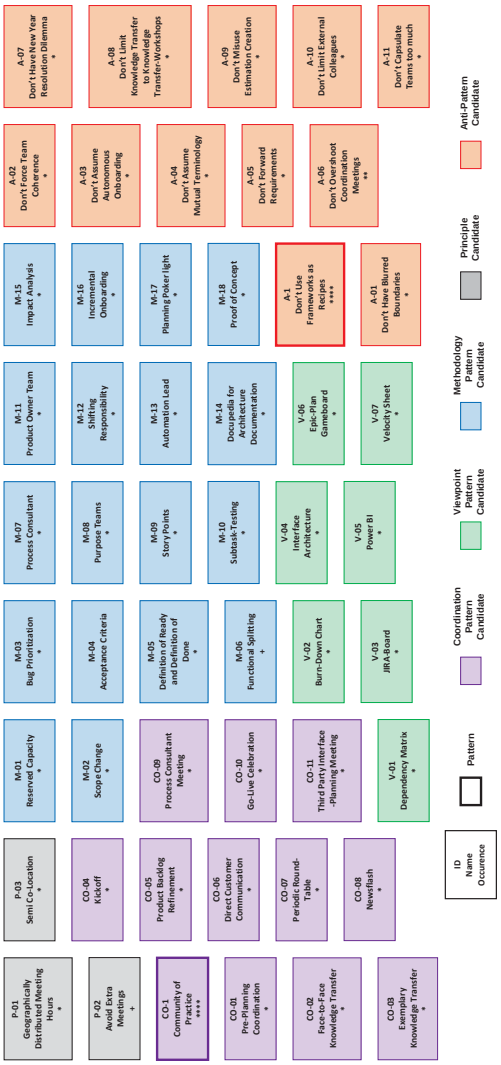


Figure 4.16.: Patterns and Pattern Candidates documented through Interviews and Observations

4.4.1. Identified Patterns

The Anti-Pattern identified at Technology LLC. was observable like a red thread through the history of the LSADP. As the Agile Transformation was basically the exact execution of the **A-1***Don't use Frameworks As Recipes* Anti-Pattern. Technology LLC.s' LSADP implemented the LeSS framework one by one in early 2017. During the Transformation Phase, the biggest Lesson Learned was to not follow a framework one by one, but rather use the frameworks provided to create your own LSADP, see Section4.2.
The other discovery was the use of **CO-1** *Community of Practice* for multiple purposes at Technology LLC. In total, there were four types of CoP active during the case study period. The types of CoP were Development, User-Interface and User-Experience, Test Automation and Scrum Master, see Figure4.6. The scale of CoP was currently on portfolio level, however, there was no limitation set on who could be part of aCoP. As the implementation of CoPs was not different to the already existing pattern, only limited adaption of the Pattern observed by Uludağ[55] was required.
The two Patterns indicate the lessons learned of the Agile Transformation at Technology LLC. creating the LSADP as a combination of individual solutions(Process Consultants, Purpose Teams),Spotify and LeSS.

4.4.2. Community of Practice*

As adapted from the documentation by Uludağ et al. [55] changed the summary, example, problems and solution to represent the ones observed at Technology LLC.. Also removing the See Also section and adding Technology LLC. to Known Uses.

CO-Pattern Overview	
ID	CO-1
Name	COMMUNITY OF PRACTICE
Alias	-
Summary	To facilitate knowledge sharing, Communities of Practice can be setup. 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 Technology LLC., the Agile Coach saw a need for inter-team knowledge share, as different methods were used in the teams.

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

Problem
Following concerns are addressed by this coordination pattern:

- **C-05** *How to facilitate shared context and knowledge?*
- **C-19** *How to deal with Internal Silos?*
- **C-39** *How to establish a Culture of Continuous Improvement?*

Forces
The following forces influence Community of Practice:

- Facilitating shared context and knowledge across the organization is difficult
- Internal silos create gaps in knowledge and communication between agile 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'[64] . Participation is not limited to a set of people but is open to everyone in the organization, who is interested in the topic [40]. The intention is to enable frequent knowledge and expertise

sharing between the participant[64] . 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 [64] . 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[40]. 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[40]. 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 [40].

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
The following Benefits have been associated with this pattern:

- Encouraging knowledge sharing for diverse topics
- Breaking up silos
- Enabling a culture of continuous improvement

The following Liabilities have been associated with this pattern:

- Requiring an active involvement of participants
- Topics in the agenda could be too diverse and broad
- Providing right incentives to the participants is challenging

Known Uses

- Technology LLC.
- Electronic GmbH
- Global Insurance Corp
- LuxCarsCorp
- Retail Corp
- Software Inc.

4.4.3. Don't use frameworks as recipes*

Adapted from Uludağ et al.[55]

Anti-Pattern Overview	
ID	A-1
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
Technology LLC. had decided to adopt agile practices at their purchasing platform team. They decide to implement the LeSS framework one-to-one.

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

Problem
Following concerns are addressed by this anti-pattern:

- **C-07** *How to deal with incorrect practices of agile development*

Forces
Following forces have been identified:

- 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 oneto-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
The following Benefits have been associated with this anti-pattern:

- All agile practices of a framework are adopted.

The following Liabilities have been associated with this anti-pattern:

- 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 , which guides an organization through adopting and implementing agile practices.

4.4.4. Good Practices

Through the interview process a total of 11 Coordination Pattern Candidates, 18 Methodology Pattern Candidates and 7 Viewpoint-Pattern Candidates were documented, see AppendixC. The pattern candidates were build up over the discovery in one interview and then building on the knowledge of a pattern candidate, the first discovered pattern candidate was incrementally changed. This process allowed to detect duplicates earlier and rather than documenting two similar pattern candidates, only the first discovered was updated with the additional information and kept as pattern candidate. By identifying so many documented Pattern Candidates, this thesis will offer an overview of all documented pattern candidates in the Appendix and only introduce a few exemplary pattern candidates for each pattern type.

Coordination Pattern Candidates

In total there were 11 Coordination Pattern Candidates documented, of these eleven, we will present two in detail and the others will be mentioned with the summary in the AppendixC. During the case study at Technology LLC., we had the chance to observe the Kickoff event of the project year 2020. In Addition, the Kickoff for Epics and larger Increments was mentioned during the interview with SM-1. Both combined generated the good coordination practice **CO-04** *Kickoff*. Kickoffs address multiple concerns, which influence all stakeholder groups at Technology LLC. Especially in distributed agile teams, coordination practices that bring teams together in physical spaces and help improve team coherence and morale. During the Kickoff event the LSADP used the opportunity to introduce some good practices observed in its own agile teams, to inspire other teams into adapting more good practices from other agile teams of the LSADP at Technology LLC..
In addition to team building games, several topics regarding Quality Assurance(automated testing and lightweight documentation) were presented and discussed. During day 2, the teams worked together to specify not only how they want to work together, but also to define their team vision for 2020 (see Figure4.17). Finally, the introduction of the good viewpoint practice **V-01** *Dependency Matrix*, addressed concerns of Knowledge Management. Overall the Kickoff offers multiple opportunities to improve the teams' morale, while simultaneously addressing categories of concern.
Another coordination good practice documented through the interview with SM-2 and PO-5 and observations is the **CO-08** *Newsflash*. The Newsflash is a good practice used at several programs at Technology LLC. and was adopted to fit the LSADP. As a platform for presenting managerial decisions, it was adapted at the LSADP to inform about Epic changes, road-map topics and anything related to the LSADP, which finds no platform in other meetings. It offers an opportunity to present the work done by the project management team(Chief Product Owner, Product Development Manager and Agile Coach) and informing the teams of potential changes or just any relevant updates to the LSADP.

Methodology Pattern Candidates

The LSADP works with **M-08** *Purpose Teams* and Process Consultants (**M-07** *Process Consultant*) to manage the domain-knowledge bottlenecks more efficiently. However, it does not mean they ignore the real threat of losing key expertise, when a PC leaves. As an attempt to address this issue, they use the good methodology practice **M-12***Shifting Responsibilities* to broaden the knowledge of each stakeholder group. Instead of simply changing teams around completely, they shift some module responsibility each year, focusing on keeping the scope small enough to not impact the working LSADP.
Through **M-12***Shifting Responsibilities* they manage to create several overlapping domain experts at an attempt to move towards working feature teams. At the beginning of the Agile Transformation, Technology LLC., tried to implement feature teams, only to realize they had teams of domain-expertise knowledge, who where very efficient in working on one domain, but lacked that prowess in other domains. Technology LLC. created a work-around for internal silos, the **M-07***Process Consultant* and **M-08***Purpose Teams* practices and implemented with **M-12***Shifting Responsibilities* a good practice as counter-measure of increasing the dependency on internal silos.
Supporting the Knowledge Management of the LSADP, the good practice of **M-14** *Docupedia for Architecture Documentation* was introduced. This is documented in the interview with DEV-3. To address the complex topic of creating lightweight documentation, Technology LLC.s' LSADP uses the arc42-template introduced by Starke and Hruschka[52]. Following the arc42-template for documenting a system architecture, the LSADP creates a lightweight architecture documentation, including the high-level module documentation, see Figure4.3.

Viewpoint Pattern Candidates

The LSADP of Technology LLC. uses a **V-06** *Epic Plan Game Board* to create a visual road map for all agile teams. This viewpoint good practice of the **V-06** *Epic Plan Game Board* creates a transparent overview of the progress of all agile teams and the epics they are trying to deliver for the year. While the **V-06** *Epic Plan Game Board* offers a good practice for communicating and coordinating visually, the good viewpoint practice **V-01** *Dependency Matrix* helps address inter-team dependencies.
As a practice applied during **CO-04** *Kickoff*, the **V-01** *Dependency Matrix* addresses Knowledge Management and Communication & Coordination concerns in one. Offering an overview for bottlenecks and reliant personnel to organize knowledge sharing. In Combination with **M-12** *Shifting Responsibilities*, the Dependency Matrix creates a clear baseline for inter-team communication and **C-85** *How to share domain knowledge across agile teams?*. In Figure4.18 the Dependencies for 2020 are visualized.

Principle Candidates

The interview process presented a total of three principle candidates, the most important documented principle candidate of this thesis is the **P-01** *Geographically Distributed Meeting Hours*. As presented in Figure4.2 all agile teams work in at least two time zones. The maximum difference of which is Germany to India, with India being four and a half hours ahead of Germany. With a working day cut in half, it is important to create guidelines for geographically distributed teams to work together.
The principle candidate **P-01** *Geographically Distributed Meeting Hours* solves the issue, by in the case of the LSADP limiting meetings with all agile team members from 8 am (Central European Standard Time) to 12 am(Central European Standard Time). Additionally time zones in which certain locations are not to be invited to meetings are included.

4.4.5. Kickoff**

CO-Pattern Overview	
ID	CO-04
Name	KICKOFF
Alias	-
Summary	Kicking off a project or year of a program. Discussing topics where LSAD wants to improve and highlighting success. Aimed at getting team motivated, informed and more comfortable with each other and the program.

Example
Technology LLC. uses Kickoff event to bring all personal of a project together. Kickoffs are used for the whole year and in special cases for the kickoff of a large project itself.

Context
Geographically distributed teams can limit the feeling of team coherence at the same time there can be several different levels of commitment to the agile process.

Problem
Following concerns are addressed by this practice:

- **C-24** :*How to create team spirit and trust amoung agile teams?*
- **C-32** :*How to deal with lacking team cohesion at different locations?*
- **C-39** :*How to create a culture of continuous improvement?*
- **C-43** :*How to enforce customer involvement?*
- **C-67** :*How to encourage development teams to talk about tasks and impediments?*
- **C-81** :*How to understand all interfaces and dependencies of the system?*

Forces
Following forces have been identified:

- Addressing culture and mindset problems is always difficult as it is hard to see actual effect take place.
- Geographical Distribution is hard to overcome by phone.

Solution
Year Kickoff (2 day event):

1. Bring all personnel to one place, whoever can't join in person should be joining via video-chat
   a) Prepare the room show achievements last year and offer all required tools for following event
   b) Plan ahead presentations and games for team building
2. Day 1 (Presentations addressing plan and goals, as well as lessons learned)
   a) Open with Management speech
   b) Present project goals and project plan (first overall and then team by team)
   c) Experts present best practices for topics important in the future of the program
   d) Between every topic make break and or team building game
3. Day 2 (Action Plan for teams)
   a) Two 90-minute sessions, first generating Working Agreements and Dependencies between Teams and then how to apply them. Presentation of result at the end of the 90 minutes.
   b) In between breaks and team building games
   c) End of Kickoff

Project Kickoff (1 Day): Similar to year kickoff instead of management involving customer and only personnel involved in the project.

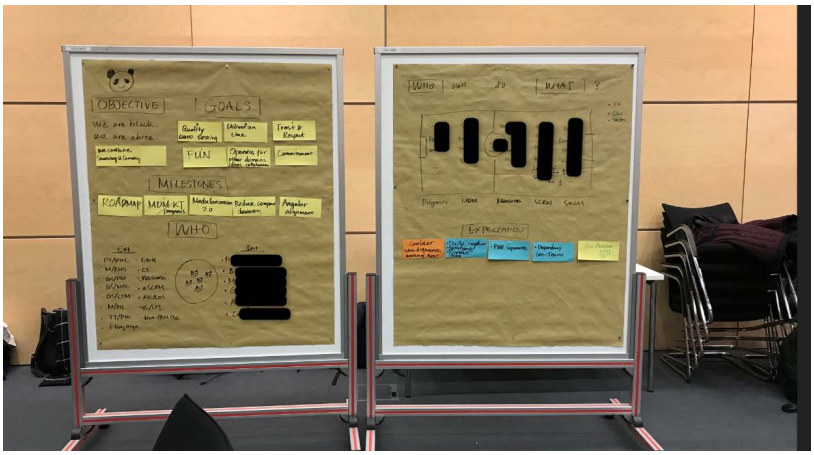


Figure 4.17.: Kickoff Team B-Setup, Module Responsibilities and Goals for 2020

Consequences
The following Benefits have been associated with this practice:

- Creates a team spirit and coherence. Get to know everyone.
- Present best practices introduces experts and makes people want to change those topics, too.

The following Liabilities have been associated with this practice:

- Time consuming.
- Expensive to bring everybody together.

See Also
V-01 *Dependency Matrix* for visualizing dependencies, which need to be addressed over the course of the project.

4.4.6. Newsflash

CO-Pattern Overview	
ID	CO-08
Name	NEWSFLASH
Alias	-
Summary	Project Management Update regarding any information of organizational nature and long-term goals affecting the Large-Scale Agile Development Program.

Example
The project management team at Technology LLC. informs all teams in a short 30 min recap of all organizational information as well as project long-term goal situations. Recurring event every third week of a month. They use the Newsflash to communicate architecture concepts and also any changes to epics or major changes affecting the agile teams.

Context
The project management team has information regarding impediments and organizational change or scope changes they want to inform the teams about. They have decisions or budget topics which are relevant for all teams and want to support a transparent communication pipeline.

Problem
Following concerns are addressed by this practice:

- **C-26** :*How to align and communicate architectural decisions?*
- **C-73** *How to deal with decreased predictability?*

Forces
Following Forces have been identified:

- Platform team mainly located in one location, they want to use a transparent and fair way of communicating any program affecting topics
- Architecture concepts from outside the project or any data security topics have to be communicate to all agile teams
- Epics or road-map changes that affect several teams need to be aligned and the same information needs to be reaching every team

Solution
Newsflash:
Set up a Newsflash Call once a month. The meeting takes place at a time, when all project relevant personnel has an opportunity to join or at least a representative of each team. The time should be restricted to half an hour.

1. Project management team discusses topics of kind:
   a) Organizational information to be shared.
   b) Architectural alignment / Solution Concepts.
   c) Changes in program road-map or epics.
2. Project management team presents agenda and includes architects and other personnel required for presentation.

Consequences
Following benefits have been associated with this practice:

- Increased transparency for work done by project management team.
- Platform for short description of architectural decisions or changes of any kind affecting everybody.
- Opportunity to acknowledge successes.

Following Liabilities have been associated with this practice:

- Additional 30 minutes of meeting.
- Information might not be relevant for all attending personnel.

4.4.7. Shifting Responsibilities

M-Pattern Overview	
ID	M-12
Name	SHIFTING RESPONSIBILITIES
Alias	-
Summary	SADP of Technology LLC shifts purpose teams and individuals responsibilities little by little over the years to create a broader pool of system experts rather than domain experts only.

Example
Technology LLC. works on a combination of newly generated and legacy systems. Actively fostering knowledge transfer, module responsibility, PO and Dev, shifts in a roughly yearly period.

Context
Complex and legacy systems are part of most development programs. Working in purpose/ feature teams hinders the knowledge sharing amongst team members. Generating internal knowledge silos.

Problem
Following concerns are addressed by this practice:

- **C-19** *How to deal with internal silos?*

Forces
Following forces have been identified:

- Complex and legacy systems are difficult to understand, when not regularly working with and on them.
- The amount of knowledge about these systems is sparse and focused in a single person or a small group.

Solution
Shifting Responsibilities:
Instead of indefinitely working on the same module, shifts of responsibilities among team members allows and automatically generates a knowledge share. Allowing for a natural growth of knowledge as team members are working on a module for a long time, instead of having to understand a module by working on it once every other month. After the period - roughly one year- team members are changing module responsibility with other team members in the same position. For the next- around a year- period, the new responsibility generates a new expert, who in case of questions can ask his predecessor.

Consequences
The following Benefits have been associated with this practice:

- Generates a Knowledge Transfer.
- Shift keeps work for individuals motivating.
- Dependency on single expert is reduced.

The following Liabilities have been associated with this practice:

- Takes a long time for a complex system to generate enough knowledge share.
- Short period every year where delivery is slower

See Also
IS combined with the **CO-1***Community of Practice* , **CO-02***Face2Face Knowledge Transfer* and **M-07***Process Consultant*

4.4.8. Docupedia for Architecture Documentation

M-Pattern Overview	
ID	M-14
Name	DOCUPEDIA FOR ARCHITECTURE DOCUMENTATION
Alias	-
Summary	Wiki with arc42 documentation of system architecture and module architecture.

Example
Technology LLC. uses the Docupedia to generate a lightweight documentation with the arc42 framework.

Context
Time constraint for documentation, as not visible business value for customers is supported.

Problem
Following concerns are addressed by this practice:

- **C-37** *How to create lightweight documentation*

Forces
Following forces have been identified:

- Difficult to explain importance of time used for documentation.
- Delivery and business value creating tasks seen as higher priority.

Solution
Docupedia for Architecture Documentation:
Docupedia hierarchy (relevant for technical documentation): Platform Documentation - Subpages –> anything directly connected to the architecture of the platform as well as platform overarching information. Module Documentation - Subpages –> all technical documentation for a specific module

Steps for documentation: Find fitting hierarchy level and create new Docupedia page.
In Case of module documentation:

- Fill in Fact Sheet (Name, Business benefit, User group, Since, Interfaces (internal), Interfaces (external), User (worldwide), Process Support, Key User Departement, Community)
- Provide Process Overview graph

- Provide System Context View (architecture graph with interfaces)
- Optionally provide FAQ, User Manual, technical detail information and automated tests

For platform documentation follow arc42[52]steps:

1. Introduction and goals (describe requirements, quality goals and stakeholders)
2. Constraints (any constraints)
3. Context and Scope (specify external interfaces)
4. Solution Strategy (Summarize decisions and solution strategies shaping architecture)
5. Building Block View (static decomposition of system)
6. Runtime View (behavior of building blocks in scenarios)
7. Deployment View (mapping building blocks to infrastructure elements)
8. Crosscutting Concepts (solution approaches view cross-cutting the system)
9. Architectural Decisions (decisions including rational)
10. Quality Requirements (high-level overview of quality goals from 1.)
11. Risks and Technical Debt (known risks and technical debt)
12. Glossary (term definitions)

Consequences
The following Benefits have been associated with this practice:

- Documentation created.
- Short and comprised overview.-Depending on creators commitment .
- Most important information with connection to detailed information.

The following Liabilities have been associated with this practice:

- Different degrees of documentation, dependent on creators commitment

4.4.9. Dependency Matrix

Through growth of the LSADP(+2 teams) at the beginning of 2020, (compare Figure4.18) there are more teams represented by columns, than were observed for this Master Thesis. The rows and final column represent the teams observed for this Master Thesis.

V-Pattern Overview	
ID	V-01
Name	DEPENDENCY MATRIX
Alias	-
Summary	During the Kickoff created, visualizing all inter-team dependenies, as well as dependencies to external teams not or organizations. Focus on Knowledge barriers and need for cooperation during development process.
V-Type	Board

Example
Technology LLC. uses the dependency matrix to visualize all dependencies of category cooperation and knowledge bottlenceks, between teams.

Context
Domain-specific teams, who shift module responsibilities or the change of personnel in teams. Creating knowledge and coordination dependencies between teams.

Problem
Following concerns are addressed by this practice:

- **C-19** :*How to deal with internal silos?*
- **C-81** :*How to understand all interfaces and dependencies of the system?*

Forces
Following forces have been identified:

- When working with a large project it can be hard to see all dependencies between different teams. Dependencies are often not obvious and can get lost.
- Communicating dependencies can be one-sided and difficult to address at the right level

Solution
Dependency Matrix:
Each team writes down dependencies to other teams on post-it notes and in the end presents those dependencies to the whole program team. The presented dependencies are then add to a matrix of all teams creating an overview(see Figure4.18and Figure4.19). In a

second step each team takes a look at all dependencies towards themselves and addresses how they will make personnel available or address these dependencies

Figure 4.18.: Dependency Matrix of inter-team dependencies for 2020 at Technology LLC.

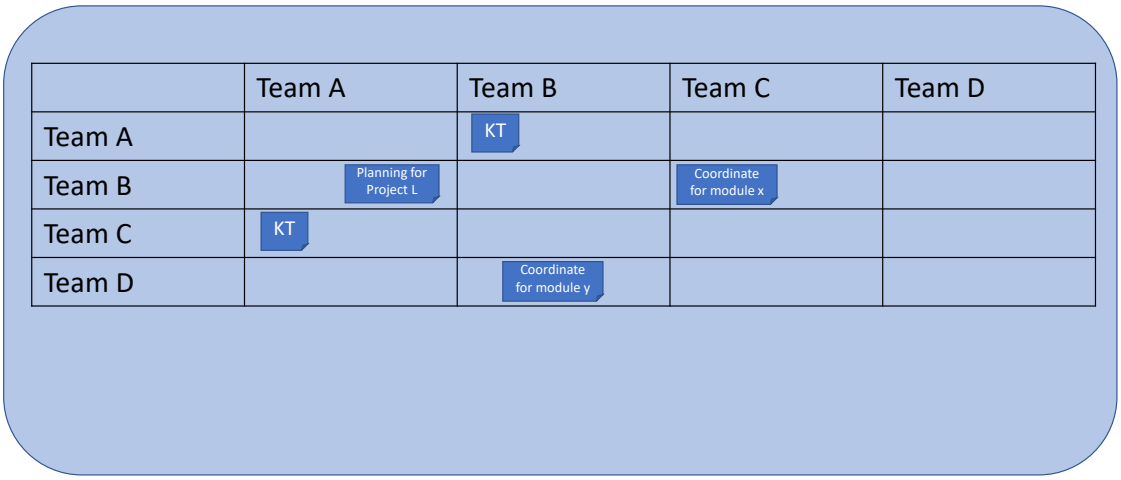


Figure 4.19.: Dependency Matrix Model of inter-team dependencies

Consequences
The following Benefits have been associated with this practice:

- Overview of all dependencies.
- Easier for planning. Makes possible bottlenecks visual.
- Common acknowledgment and cooperation plan able.

The following Liabilities have been associated with this practice:

- All personnel needs to be involved otherwise not reliable-Time consuming, when not accurate

See Also
Gets created during the **CO-04***Kickoff* event.

4.4.10. Epic Plan Game Board

V-Pattern Overview	
ID	V-06
Name	EPIC PLAN GAME BOARD
Alias	-
Summary	Creating a traceable overview for all Epics to be done during the year by taking Epics and organizing them on a Game Board. Creating a nice visual for tracking the process and allocation of the epics.
V-Type	Board

Example
Technology LLC. works with an epic plan in a large-scale agile development program. The epic plan is relevant for a single year and constructed at the end of a year. Presented in the kickoff it depicts all epics planned for each team, usually depicted in a poster and uploaded to the wiki.

Context
Whenever working with a budget-planning horizon of a limited time. Requiring the agile program to first allocate their capacities according to the available budget, most easily done by planning epics for the planning horizon.

Problem
Following concerns are addressed by this practice:

- **C-22** :*How to balance short-term and long-term goals?*
- **C-78** :*How to synchronize sprints in the large-scale agile development program?*
- **C-80** :*How to manage overarching backlog item prioritization with multiple product owners?*
- **C-83** :*How to manage requirement development for multiple teams?*

Forces
Following forces have been identified:

- Budget limits the capability of agile program as well as the planning horizon.
- When coordinating several customers as well as working with limit capacities, it is impossible to work simultaneously on projects for all customers as only a certain amount of development capacities are available.

- Creating a quantifiable goal out of a large amount of requirements from different customers

Solution
Epic Plan Game Board:
Each Product Owner coordinates with his customers what they want to have added and what needs changing. The headlines of these meetings summarize an epic, the topic of the requirements. The epic describes on a very abstract level, what all requirements relevant for this topic create or change. All these Epics come with a budget, according to this budget; the agile program can now estimate how much capacities are required. From all capacities assigned to the epic plan, the program can now allocate epics to the teams. The epic plan stands, when all team capacities for the planning horizon have been coordinated and allocated to a sprint during this period.(compare Figure4.20 and Figure4.21) During the finalization of the epic plan, eventual conflicts of personnel capacities become visual and can be avoided or limited.

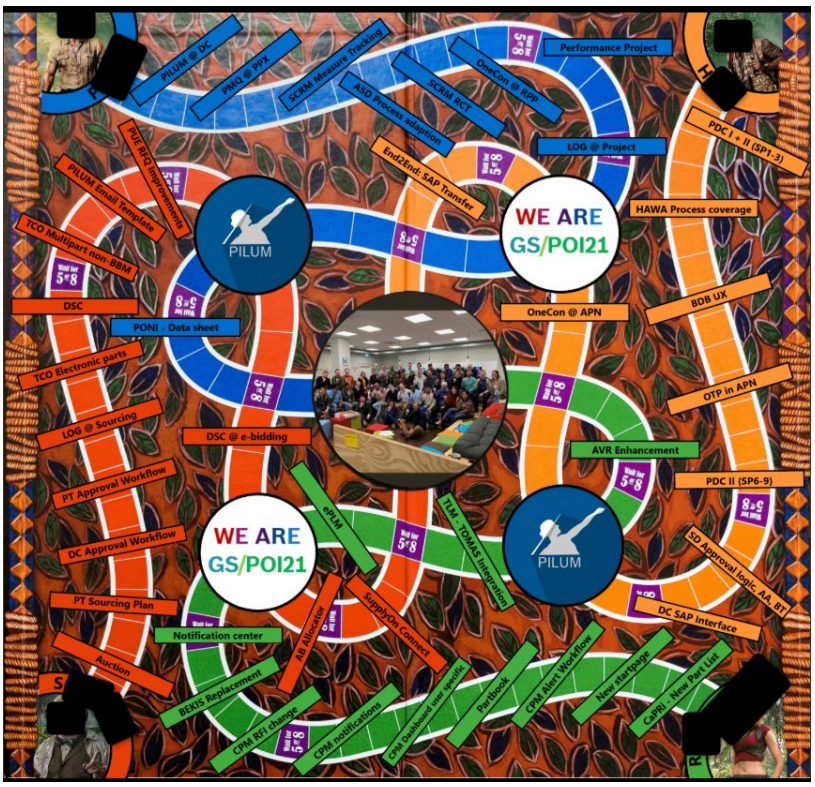


Figure 4.20.: Epic Plan Game Board for 2019 at Technology LLC.

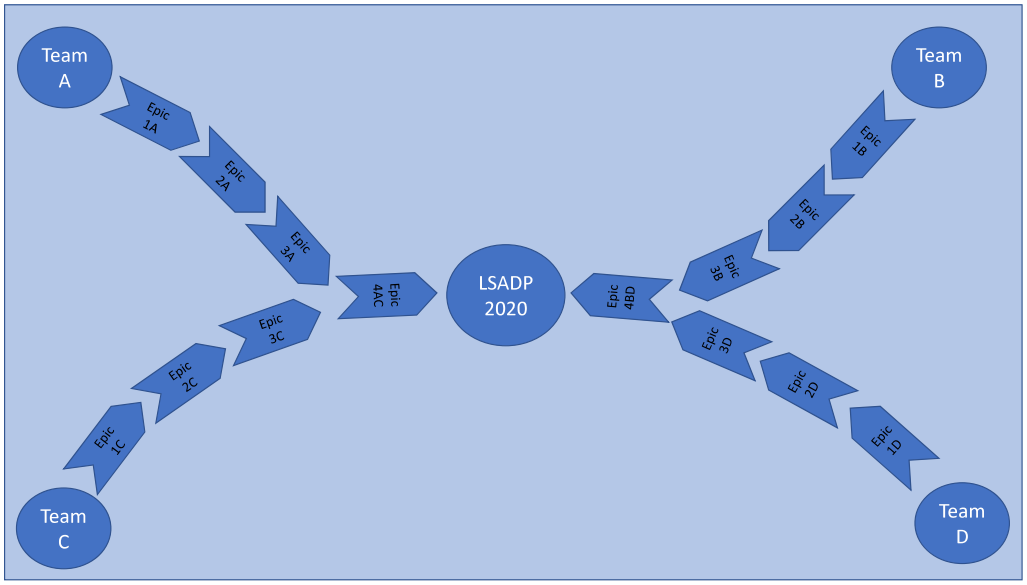


Figure 4.21.: Epic Plan Game Board Model

Consequences
The following Benefits have been associated with this practice:

- Overview of planning horizon .
- Success measure through finishing an epic.
- Team members capacities are respected.

The following Liabilities have been associated with this practice:

- Loss of flexibility

See Also
Usually used in combination with **V-05** *Power BI* and **V-07** *Velocity Sheet*.

4.4.11. Geographically Distributed Meeting Hours

Principle Overview	
ID	P-01
Name	GEOGRAPHICALLY DISTRIBUTED MEETING HOURS
Alias	-
Summary	Technology LLC. creates for each project a set of geographically distributed working hours. With this method, they define the working hours, when all locations of the project are working simultaneously. Hence, a time span in which meetings can be organized are set for the project or teams of the project.
Type	Coordination
Binding Nature	Required

Example
Technology LLC. has geographically distributed teams in different time zones, working in a large-scale agile development program. Communication and coordination relies on using daily and weekly repeating meetings. Additionally individuals of a team have a need to coordinate and all team members prefer direct communication.

Context
Teams coordinate in Daily's and Product Backlog refinement meetings handle requirement refinements. These repeating meetings during a sprint are most efficient when attended by all team members. The team, however, is geographically distributed and has different working hours due to different time zones.

Problem
Following concerns are addressed by this principle:

- **C-03** :*How to coordinate geographically distributed agile teams?*
- **C-29** :*How to facilitate agile teams to participate at cross-shore meetings?*
- **C-30** :*How to synchronize working hours of cross-shore agile teams?*

Forces
Following forces have been identified:

- Organizational guideline to use both onshore and offshore employees in software development for projects over a certain size.
- Employees cannot work at exactly the same time all day, when working in different time zones.

- Knowledge among employees as well as the complexity of the system block the program from creating co-located teams.

Solution
As teams should consists of both On- and Offshore in larger projects, a guideline supporting this circumstance is defined as follows: Therefore, defining limitations of possible general working hours for each time zone involved. Communicating these limitations by involving them in the working agreements of the large-scale development program. Focusing on including these limitations in all individual working agreements of teams. Forcing everyone to be mindful of the multiple time zones involved in a project. Creating a schedule for meetings where the entire program team is required to be attending and therefore creating a skeleton for the individual teams to coordinate their weekly and daily repeating meetings around **Known Uses**
Technology LLC.

Consequences
The following Benefits have been associated with this principle:

- Clear rules for communication and coordination efforts.

The following Liabilities have been associated with this principle:

- Limit time for meetings. Depending on locations overlap can be really low

See Also
CO-04 *Kickoff* is used for agreeing on the distributed meeting hours.

4.4.12. Bad Practices

The LSADP of Technology LLC. faces some bad practices, which have to be avoided. In the interview with PO-3, we documented the bad practice **A-04** *Don't assume mutual Terminology*, an anti-pattern candidate, which later was updated through the interviews with PO-5 and PO-6. When working with customers of external or internal traditional teams, the POs faced the issue of using common terminology for the LSADP only to be misunderstood by their counter-part. To address this misunderstanding, PO-3 started to focus on explaining his terminology when meeting customers, who aren't as familiar with the agile practices.
Another bad practice documented at the LSADP of Technology LLC. was **A-07** *Don't Have New Years Resolution Dilemma*. While practices like **M-05***Definition of Ready and Definition of Done* and **CO-01** *Pre-Planning Coordination* were introduced at the LSADP to avoid **A-06** *Don't overshoot coordination meetings* and **A-09** *Don't misuse estimation creation*.
The bad practice of teams committing to **M-05***Definition of Ready and Definition of Done* and not actually adhering to the practice was observed. Most of the interviewees in some kind used the **A-07** *Don't Have New Years Resolution Dilemma* bad practice. Irrelevant of stakeholder group, the agile teams tended to commit to a new way of solving an issue or documenting a process, only to fall back on the old process. While the LSADP works fine, the wish of *"committing to commitments "Product Owner(PO-1) (2019)* was mentioned by several interview partners and is in direct correctional with the bad practice and anti-pattern candidate **A-07** *Don't Have New Years Resolution Dilemma*.

4.4.13. Don't assume mutual Terminology Understanding

Anti-Pattern Overview	
ID	A-04
Name	DON'T ASSUME MUTUAL TERMINOLOGY UNDERSTANDING
Alias	-
Summary	When coordinating with traditional teams, Product Owners use terminology used in the agile teams and don't explain the terms. Leading to misunderstandings and possible misinformation being spread.

Example
Technology LLC. has a large-scale agile development program and several traditional programs, when working on a same module the communication and especially when using terminology describing steps in the process or any LSAD-specific terminology.

Context
A Development Process involving teams working in traditional (i.e. Waterfall, V-Modell, etc.) frameworks for software development and teams working in agile (i.e. SAFe, LeSS, etc.) frameworks.

Problem
Following concerns are addressed by this bad practice:

- **C-20** :*How to facilitate communication between agile teams and other teams using traditional practices?*

Forces
Following forces have been identified:

- Terminology in traditional and agile processes is different yet often addresses similar concepts.
- Only because people work together does not mean they have the same terminology understanding.

General Form
Traditional and agile teams working together in large project or on large module. Additionally observable when communicating with customers. When describing an Product Increment or talking about the development process in general and referring to terminology used within the LSAD program. The customer or project partner does not have the same definition for the term used and assumes a different outcome. Or just assumes a

negative connotation for terms(i.e. MVP is seen as a low value product).

Consequences
The following Benefits have been associated with this bad practice:

- working with an agile team with terminology that appears different may leave a positive connoted feeling towards agile team.

The following Liabilities have been associated with this bad practice:

- Misunderstanding between Teams especially focused on priorities and the therefore described order of which features are developed. While in traditional programming there is little to none prioritizing.
- When talking about MVP's teams in traditional working programs assume they only get a prototype or something of less value. While MVP is used to figure out the minimum requirements to get a working platform or feature.

Revised Solution
Use the **CO-07** *Periodic Round-Table* to clarify terminology and build up a common understanding.

4.4.14. Don't have New Year Resolution Dilemma

Anti-Pattern Overview	
ID	A-07
Name	Don't have New Year Resolution Dilemma
Alias	-
Summary	LSAD aims at introducing artifacts to address some concerns and commits to implementing them, however, the teams don't go through with their commitment and the concerns remain.

Example
Technology LLC.'s LSADP wants to focus more on automated testing, they want improve the overall quality as unified decision and goal.

Context
Delivery pressure and budget dependency from delivering features.

Problem
Following concerns are addressed by this bad practice:

- **C-36** *How to establish automated testing?*
- **C-79** *How to balance amount and quality of delivered Requirements?*

Forces
Following forces have been identified:

- Delivery-driven program, missing infrastructure in already existing unit test.
- Missing documentation on Tests limits possibility to create automated testing when working under delivery pressure. Yearly allocated budgets, limit flexibility.

General Form
Established guidelines like Working Agreements, DoR and DoD crumble under delivery-pressure. First the threshold for achieving these guidelines and agreements gets lowered and maybe even completely ignored to deliver a feature on time or with as little delay as possible. Similar to new year resolution as soon as the pressure increases, resolutions and agreements are set aside.

Consequences
The following Benefits have been associated with this bad practice:

- Delivery speed is increased.

The following Liabilities have been associated with this bad practice:

- Inferior quality in delivered products
- Misues of agile practices as there is no commitment to the correct application.

Revised Solution
Committing to commitments. Start distributing technical sprint topics over the whole year, resolving them as part of sprint.

- focusing more on quality assurance within a year.
- Start distributing non-functional requirements and concept implementations into the year.

Sprint Planning only Team no PO, the SM takes over the PO presentation to release pressure on teams commitment. **CO-01** *Pre-Planning Coordination* within Team to coordinate the capacity limit they are willing to commit. Backlog is estimated and prioritized before sprint planning!

See Also
Plan ahead in the units of DEV and PO with **CO-01** *Pre-Planning Coordination*,

4.4.15. Mapping of Concerns and Pattern Candidates

With the identified concerns and the documented pattern candidates and two patterns, it is interesting to take a look at how these are connected. The figures below represent the Mapping of recurring concerns to good or bad practices documented for the same stakeholder group.

Mapping Development Team

Beginning with the stakeholder group of DEV (see Figure4.22) we can identify a clear influence of the newly introduced roles of **M-07** *Process Consultant* and **M-13** *Automation Lead*. These two pattern candidates address concern of the category Quality Assurance and Knowledge Management, like **C-36** *How to establish automated testing?* and **C-85** *How to share domain knowledge across agile teams?*.
This thesis tried to address the concern category of Quality Assurance , as it represents 40% of all DEV concerns by instantiating a pattern observed at another organization, in Section4.5. Additionally, the introduction of **M-14** *Definition of Ready and Definition of Done* aims at resolving the concern **C-37** *How to create lightweight documentation?*, another Knowledge Management concern.
The effect of **A-11** *Don't encapsulate teams too much* is also quite impressive, as there are four good practices, which try to resolve the encapsulated team construct, the LSADP created through their Agile Transformation and the introduction of **M-08** *Purpose Teams* and **M-07** *Process Consultant*.

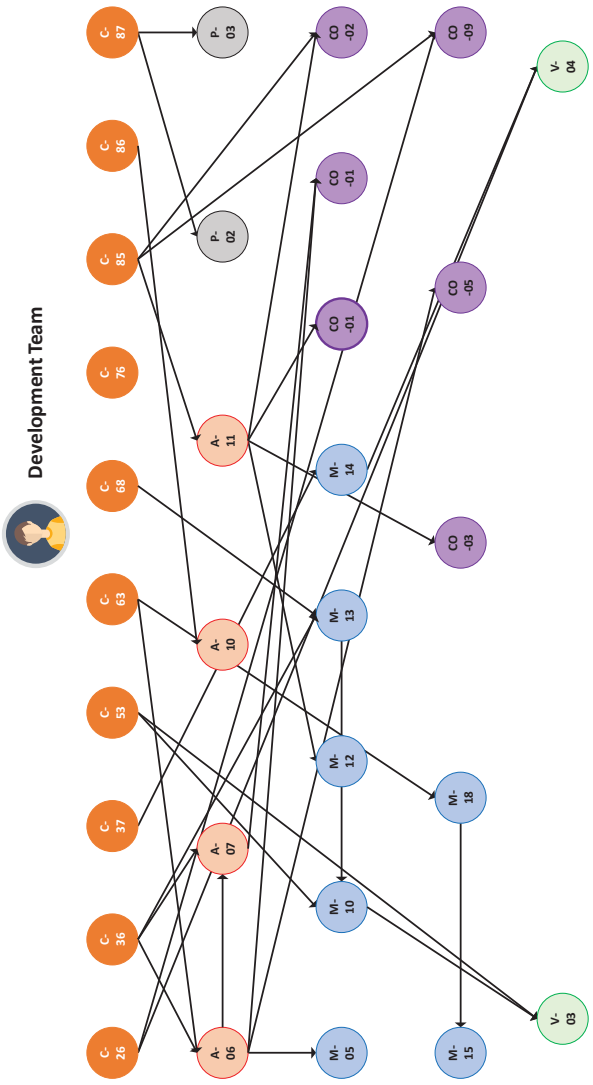


Figure 4.22.: Mapping Concerns and Pattern Candidates of Development Team

Mapping Product Owner

The mapping for the PO stakeholder group(see Figure4.23) is revealing a lot of connections between concerns identified and good and bad practices documented.
With **C-80** *How to manage overarching backlog item prioritization with multiple product owners?* being a major concern addressed by a total of six good practices. The newly observed situation of **M-08** *Purpose Teams* in combination with **M-11** *Product Owner Team* introduces the concern **C-80** *How to manage overarching backlog item prioritization with multiple product owners?* to the LSADP.
However, the LSADP identified this concern and tried to resolve it with visualizing the epics **V-06** *Epic Plan Game Board*, offering additional information during the sprint for the POs **V-05** *Power BI* and displaying inter-team dependencies with the introduced **V-01***Dependency Matrix*.
Additionally, interesting to observe is the connection of the three bad practices **A-01** *Don't Have Blurred Boundaries Requirements Engineering*, **A-03** *Don't assume Autonomous On-Boarding* and **A-05** *Don't forward Requirements* to **C-10** *How to create precise requirement specifications for the development team?* and **C-15** *How to elicit and refine requirements of end users?*, which are two major Requirements Engineering concerns. These three bad practices have not be addressed sufficiently and offered an opportunity to instantiate a pattern (see Section4.5) at Technology LLC.

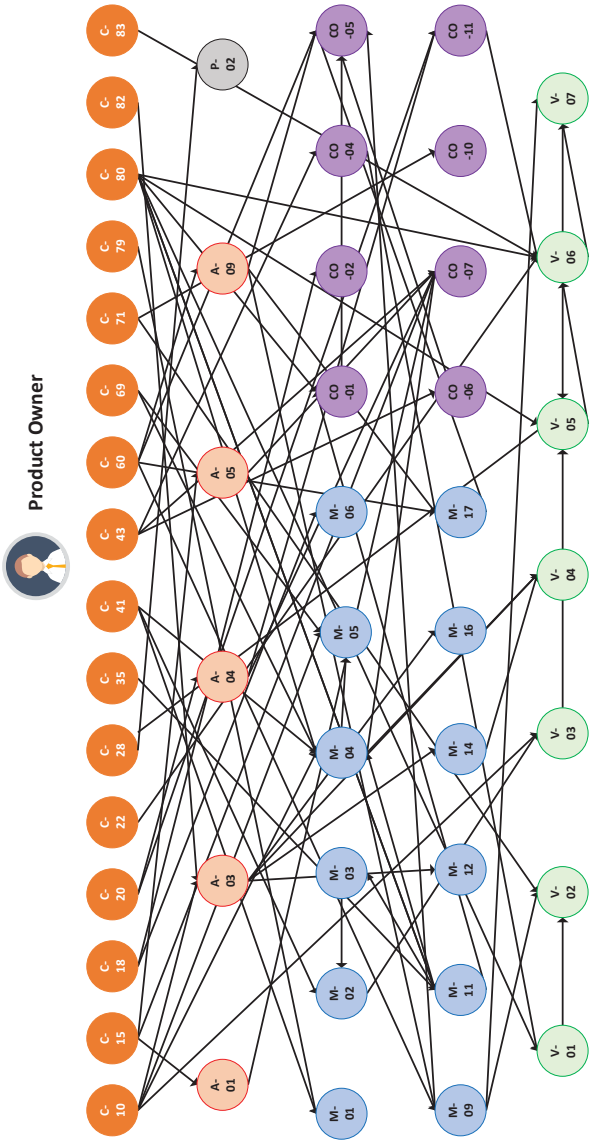


Figure 4.23.: Mapping Concerns and Pattern Candidates of Product Owners

Mapping Scrum Master

Finally, the mapping of the SM stakeholder group reveals the effect Internal Silos **C-19** *How to deal with internal silos* have on the LSADP of Technology LLC. By intentionally creating a one-stop contact person for knowledge sharing in the Purpose Teams(**M-08***Purpose Team*), with the Process Consultant (**M-07** *Process Consultant*) as exactly that bottleneck with a 50% role for addressing knowledge sharing, the LSADP might have resolved to some extent their knowledge management issue in the short-term horizon.
However, by introducing **M-12** *Shifting Responsibilities* and **CO-1** *Community of Practice* they aim to resolve Knowledge Management concerns in a long-term horizon. The recurring concerns of the Culture & Mindset category are with 34% the main concern category of the SM stakeholder group at Technology LLC., however, the Mapping (see Figure4.24) reveals that only five (**C-24**, **C-33**, **C-39**, **C-67**, **C-84**) of these 12 concerns are actually addressed. Creating an opportunity for this thesis to try and address some more of the Culture & Mindset concerns of the SM stakeholder group with instantiating patterns from other organizations as described in Section4.5.

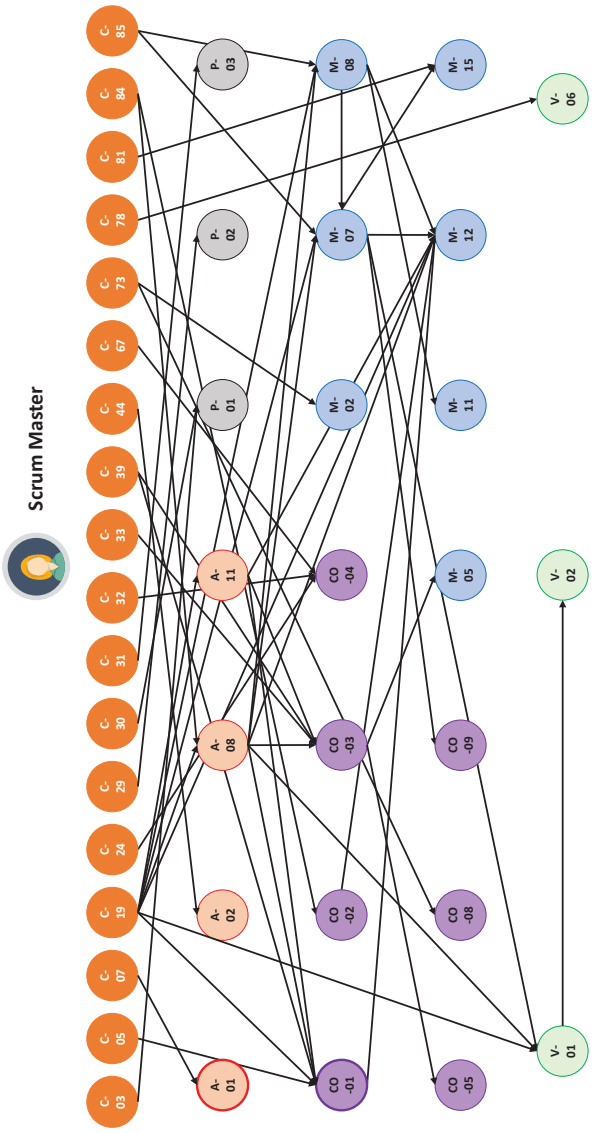


Figure 4.24.: Mapping Concerns and Pattern Candidates of Scrum Masters

4.5. Implementation of Pattern and Lessons Learned

As the final part of the case study at Technology LLC., we instantiated one of the Coordination Patterns documented by Uludağ et al.[55].
During the case study we identified a total of 65 concerns and documented 38 pattern candidates aimed at resolving these concerns together with one coordination pattern **CO-1** *Community of Practice* and 11 anti-pattern candidates together with one anti-pattern **A-1***Don't use Frameworks as Recipe*.
As summarized in Section4.4.15 each stakeholder group had a concern category indicating potential for improvement. These concern categories are:

- Quality Assurance for concerns of DEV
- Knowledge Management and Requirements Engineering for PO
- Culture & Mindset for SM

As already mentioned, we were able to participate and observe the LSADP Kickoff event. In the process of the Kickoff event, the presentations of good practices from the LSADP were focused on Knowledge Management, Quality Assurance and the event itself was aimed at **C-39**creating a culture of continuous improvement. Following the Pattern-based Design Research approach by Buckl et al.[10], we then proposed six solution designs of already observed patterns (see Section4.5.1) from the pattern catalog[61] to the LSADP of Technology LLC.. The solution design patterns were presented to all interview partners on two separate dates.

4.5.1. Patterns Provided

The patterns documented here were presented to the stakeholder groups at Technology LLC. with the **CO-1** *Community of Practice for Product Owners-instantiated solution* being instantiated as a result. The patterns not documented in detail in this section are recorded in the AppendixC.6.
Quality Assurance solutions for DEV
For the DEV stakeholder group we presented two patterns aimed at supporting the Quality Assurance concerns identified in Section4.3.2. **CO-3** *Communicating the Architecture*, observed by Uludağ et al.[58]. Is an approach aimed at communicating architecture and sharing the vision for the architecture and how to achieve architectural goals. The idea is to save time-slots in LSAD-wide events for architects to present topics or progress of architectural alignment. Additionally, we advised to even further encourage the Process Consultant of the Purpose Teams to address architecture topics in team events.
With this pattern we aimed at addressing the concern **C-26** *How to align and communicate architectural decisions?* of the DEV.
The other approach presented to address the Quality Assurance concerns of DEV, was

M-1*Quality Gates*, observed in Uludağ and Matthes[60]. The concept of introducing Quality Gates is a measure introducing manual gates in the pipeline. A Gate is defined by an architect using a tool like SonarQube in Jenkins setting conditions which are constantly checked. When a Gate is triggered, a manual pull by a dedicated developer is required to move the code along in the pipeline.

Knowledge Management and Requirements Engineering solutions for PO

For the LSADP at the case study partner, we used the CoP pattern documented (see Section4.4.2) to represent the configured design aimed at the LSADP of Technology LLC. for the instantiated solution see Section4.5.4 and the deviations from the introduced **CO-1** *Community of Practice* by Uludağ et al.[55],see Section4.5.5. The finalized **CO-1** solution design is presented in Section4.5.4.

Additionally, we presented the **CO-2** *Supervision*, documented by Uludağ and Matthes[59] coordination pattern to address Knowledge Management and Requirements Engineering concerns of the stakeholder group PO.

CO-2 *Supervision* is a Coordination pattern observed at several organizations. During a *Supervision* a stakeholder group meets up and discusses two selected problems faced by a participant of the **CO-2** *Supervision*. Supervisions create an open forum in which solution approaches are discussed, dedicated to real-life problem situations a participant faced. Through the discussion, all participants see different solution approaches, which they could apply when facing a similar problem situation.

Culture & Mindset solutions for SM

During the identification phase, we observed 34% of SM concerns to be of the Category Culture & Mindset. As the LSADP is delivery-driven and works on multiple epics at a time, we advised the use of the **P-1** *Celebrate every Success* principle documented by Uludağ et al.[55]. Additionally, we presented the practice of **CO-4** *DDD: Event Storming Workshops*, observed by Uludağ et al.[56], to the LSADP. In an Event Storming Workshop, the whole development team and architect visualize all events of a module and discuss architectural solution concepts which could be applied. The goal of **CO-4** *DDD: Event Storming Workshops*is to both introduce architectural considerations on the team level, while also establishing and fostering the continuous improvement of a LSAD.

CO-1 :*Community of Practice*

4.5.2. Celebrate Every Success **

As adapted from the documentation by Uludağ et al. [59]

Principle Overview	
ID	P-1
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
Following concerns are addressed by this principle:

- **C-24** :*How to create team spirit and trust among agile teams?*
- **C-33** :*How to build trust of stakeholders in agile practices?*

Forces
Following forces have been identified:

- 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
Following Benefits have been associated with this principle:

- 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.

Following Liabilities have been associated with this principle:

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

See Also
This principle can be used in combination with **CO-08** *Newsflash*.

Known Uses
This principle is used by:

- Autonomous Cars Group
- AgileConsultants GmbH
- Retail Corp

4.5.3. Presentation of Provided Patterns

On the first date, the six solution designs were presented to my advisor at Technology LLC. and two of the three project management group members. All three attendees were part of the interview process, namely the PO-4,PO-5 and SM-2. Again the PO-4 and SM-2 make up two thirds of the project management group (Chief Product Owner and Product Development Manager). During this presentation the solution designs were well received and together we decided to present the solution designs on a second date to the interview partners and let them decide, which solution designs they would like to instantiated. On the second date, all interview partners were invited to join the presentation of the solution designs, as they were already familiar with the purpose of this case study.
However, three other program members joined in on the presentation. The three members, were one PO, one dual-student working with the PO Team and the IT-Architect. The presentation of the solution designs was provided in the following steps:

1. Summary documented pattern candidates and identified concerns
2. Focus on the identified concern categories
    - Amount of identified concerns for each category and documented pattern candidates addressing these concerns
    - Recall of some bad practices documented during the interviews (see Figure4.25)
    - Short break for any Questions
3. Presentation of solution designs
    - Which concerns does the solution design address?
    - How often were these concerns identified during interview process?
    - Which Benefits were observed at other organizations with these solution designs?
    - How the solution design could be implemented?
4. Discussion

Overall, the presented solution designs were once more well received and the stakeholders started asking detailed questions about the solution designs, The highest interest came from the PO group, as they have not thought about discussing their concerns in an open forum and documenting that discussion, which was the basis of the two solution designs presented to them. While **CO-2** *Supervision* was well received, the length and complexity of that practice was too much for the PO stakeholders.
However, the **CO-1** *Community of Practice* offered similar benefits as the **CO-2** *Supervision*, while being less time consuming and more controllable from the perspective of the PO stakeholders.
Interestingly, the POs were able to see the benefits of **CO-1** *Community of Practice* in their

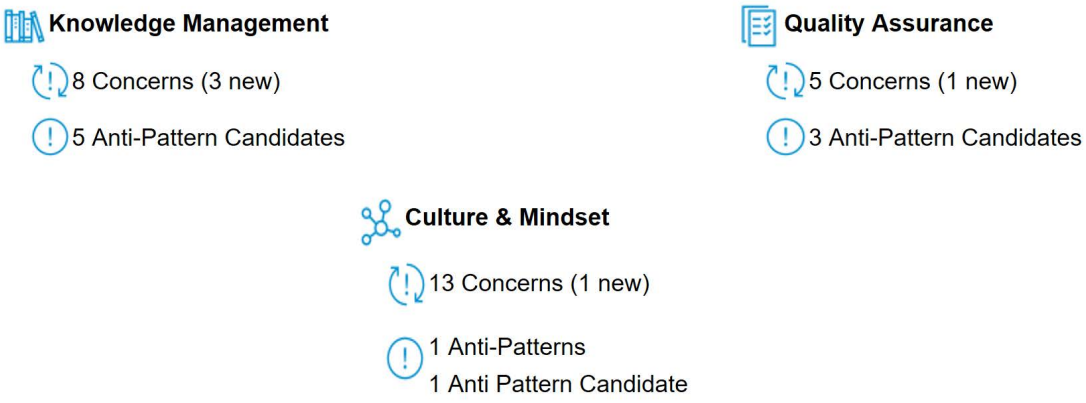
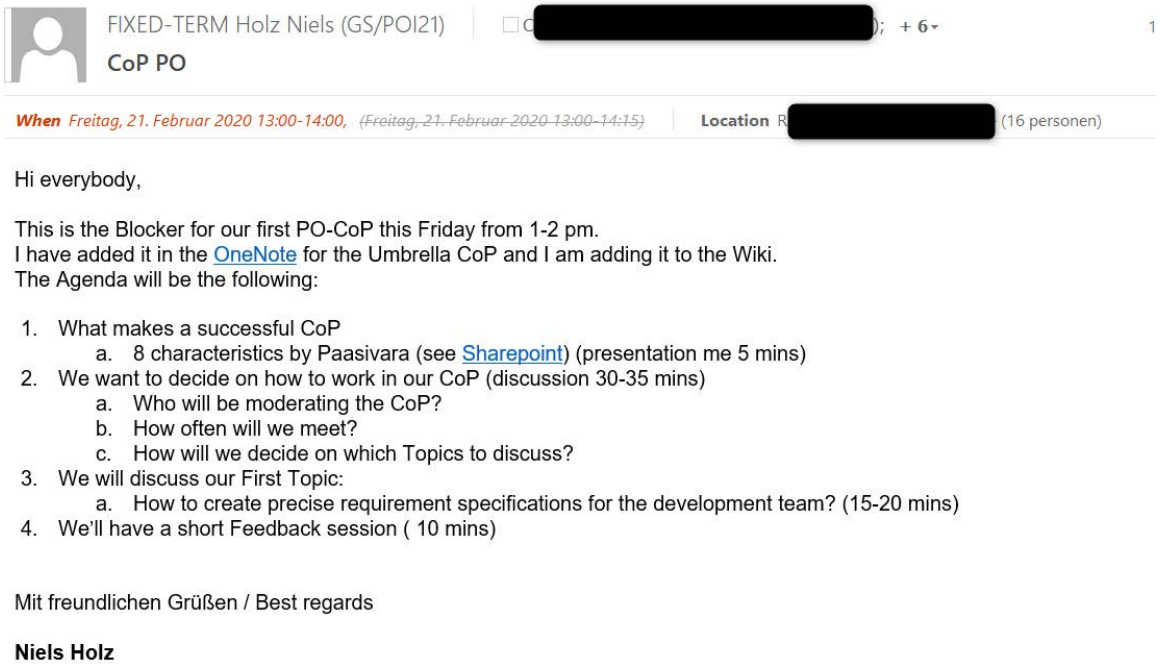


Figure 4.25.: Concern Categories for Implementation Presentation with bad practices

own LSADP with the Development, Test, UI/UX and SM CoPs being already in place, nonetheless, they never assumed a similar concept would be useful for their responsibility area. As of now, most information about the application of CoP is focused on development and technical background benefits.
The eight characteristics of a successful CoP introduced by Paasivara[40], never directly connect Community of Practice to development topics. A successful CoP has interesting topics, to whom these topics are interesting is never specified. While the CoP for SM introduces the benefits of CoPs to not strictly technical stakeholders,the SMs at the LSADP are associated and involved in the development process. By focusing on the benefits a CoP for POs could produce and how many concerns would be addressed, the POs were convinced to instantiated their own CoP.
After the Discussion following the presentation, the attendees of the stakeholder group PO decided to instantiate the solution design of a **CO-1** *Community of Practice* with our guidance for the implementation. Additionally to instantiating the **CO-1** *Community of Practice for Product Owners - instantiated Solution*, the attendees of the first date committed to implementing the **P-1** *Celebrate every Success* principle by including it in the **CO-08**. Final observation from the second date was the IT-Architects continued interest in **CO-4** *DDD: Event Storming Workshop*, however, by the end of this thesis, no commitment or implementation attempt was documented.

4.5.4. Implementation of Community of Practice for Product Owners

The Implementation of the the Community of Practice, was organized along the eight characteristics of a successful CoP identified by Paasivaara[40] and documented in the work of Uludağ et al[55]. At Technology LLC. we instantiated a Community of Practice and not an Empowered Community of Practice[55]. The difference between those two types lies in the applicability of decisions made in the Community. While the LSADP of Technology LLC. doesn't limit the decision making power of a CoP, there is also no mandatory application of practices designed or decisions made of an CoP. Hence, the instantiated **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated solution* does not apply to be an empowered CoP and was not supposed to be. In Coordination with the



FIXED-TERM Holz Niels (GS/POI21) [redacted]; + 6

CoP PO

When Freitag, 21. Februar 2020 13:00-14:00, ~~(Freitag, 21. Februar 2020 13:00-14:15)~~ **Location** R[redacted] (16 personen)

Hi everybody,

This is the Blocker for our first PO-CoP this Friday from 1-2 pm.
I have added it in the OneNote for the Umbrella CoP and I am adding it to the Wiki.
The Agenda will be the following:

1. What makes a successful CoP
   a. 8 characteristics by Paasivara (see Sharepoint) (presentation me 5 mins)
2. We want to decide on how to work in our CoP (discussion 30-35 mins)
   a. Who will be moderating the CoP?
   b. How often will we meet?
   c. How will we decide on which Topics to discuss?
3. We will discuss our First Topic:
   a. How to create precise requirement specifications for the development team? (15-20 mins)
4. We'll have a short Feedback session (10 mins)

Mit freundlichen Grüßen / Best regards

Niels Holz

Figure 4.26.: Community of Practice for Product Owners - invitation and Agenda

other documented CoPs at the LSADP, we introduced the CoP in the similar fashion by following these steps:

- Create meaningful Agenda(see Figure4.26) and invite POs of LSADP, present at one of the two presentation dates
- Make it accessible for everyone, who wants to join (Skype-for-Business conference room and physical conference room)
- Create online presence for CoP

1. OneNote Structure (see Figure4.27) and Documentation Pages
2. Open Topic Collection

- Moderate first CoP

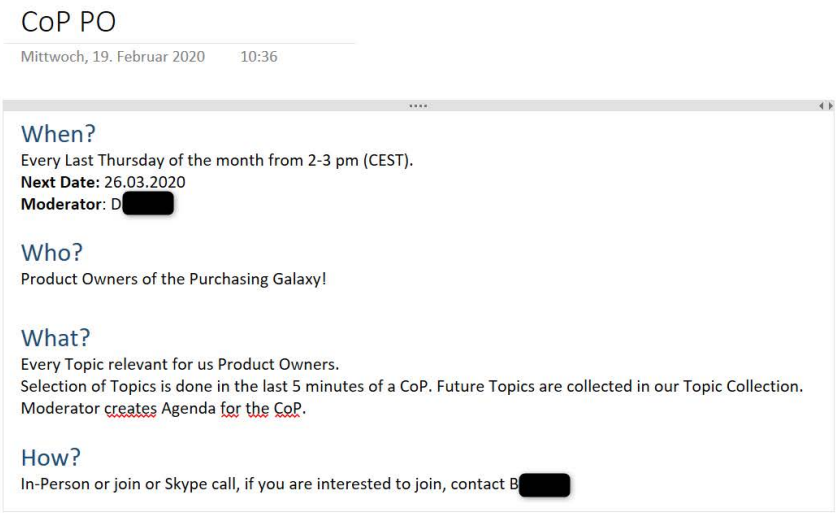


Figure 4.27.: Community of Practice OneNote Homepage

At the event of the first CoP, we started by presenting the eight characteristics of a successful CoP.
The characteristics of a successful CoP are **Interesting topics**, **Passionate Leader**, **Proper Agenda**, **Decision Making Authority**, **Open Community**, **Supporting tools to create transparency**, **Suitable rhythm** and **Enabling Cross-Location participation**[40].
For ease of implementation, we then used the second Agenda point to decide on the rhythm, leader/moderator and how to assign the topics discussed (see Figure4.26).The actual documentation of the CoP is presented in the two figures, Figure4.28 and Figure4.29.

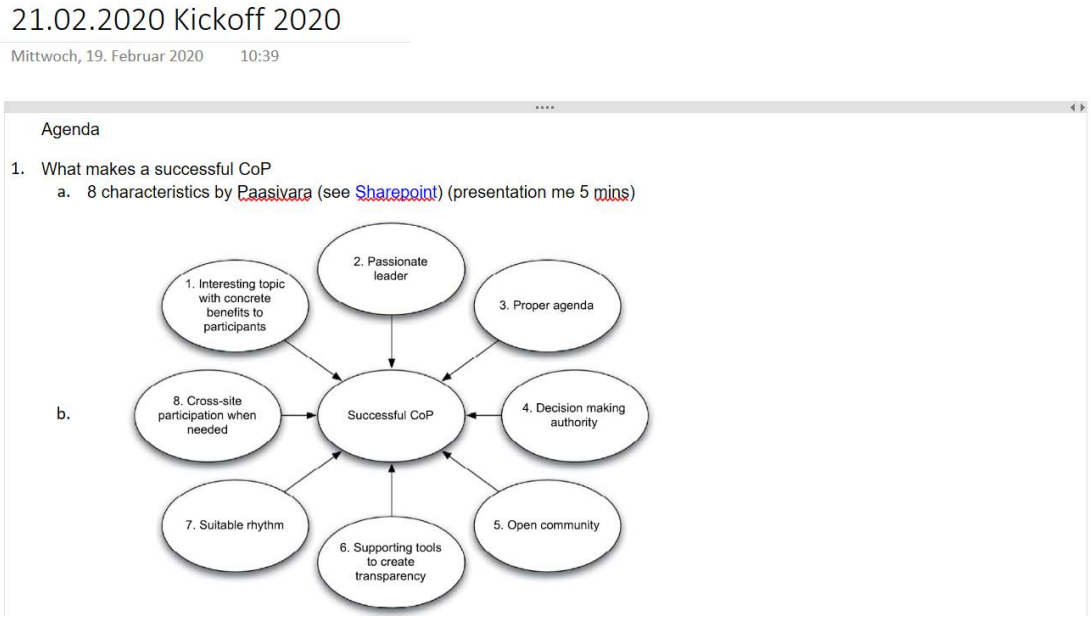


Figure 4.28.: Screenshot of instantiated Community of Practice for Product Owners - Documentation

In Adoption to the three week Sprint of the LSADP, the attendees decided to hold an one hour long CoP every last Thursday of the month. The Attendees decided, they would use the initial group of five POs present as a leader group for the CoP going forward. Deciding at the end of each CoP, which topics to discuss next and who would moderate the next CoP. The moderator of a CoP instance is responsible for creating the agenda as was decided by the attendees. One exception of the leader group was one attendee volunteering to maintain the recurring event in the calendar and create a mailing-list.

As final part of this agenda point, they decided to use the provided open topic collection and as already mentioned decide in the last 5 minutes of a CoP, which topic will be discussed during the next CoP. While the CoP wanted to have a transparent documentation of their discussion, they decided to start the CoP by only including POs, who are part of the LSADP.
This decision was made, because they want to use the CoP in the beginning to address the **C-81** *How to understand all interfaces and dependencies of the system?* concern identified during the interview process by all POs participating. To address this concern they planned on discussing one POs module per CoP and then discuss in addition one other topic of concern, not related to Knowledge Management. Further, we used the first CoP to discuss **C-10** *How to create precise Requirement Specifications for the Development Team?*. For the discussion we used an exemplary requirement provided by one PO, who was unhappy with how long it took for the Development Team to understand his requirement and wanted to know, how the others would have specified the requirement.

b. Who will be moderating the CoP?
   a. Rotating moderation deciding at the end of a CoP meeting, who will moderate the next CoP.(5 min timeslot for planning next CoP and Moderation)
   b. Agenda is generated by moderator of next CoP
c. How often will we meet?
   i. Last Thursday of each month. 1h Blocker from 2-3 pm
d. How will we decide on which Topics to discuss?
   a. When topics are already present, decision at the end of CoP which topics will be discussed.
   b. OneNote for all topics and documentation of meetings. Startpage for Topic collection and general information
e. For now only Purchasing Galaxy PO as members, invitation via mail. B[redacted]as initiator for creating meeting in calendar
f. Meeting participation
   a. Skype Meeting and meeting rooms as available
g. Decision making enabled by default

3. We will discuss our First Topic:
   a. How to create precise requirement specifications for the development team?
      a. CPREQ-25606
         i. Split requirements if possible, try not to have more than 5 d development days in one ticket.
         ii. Use subtasks and encourage developers to tell you when requirement is too large or complex.
         iii. When having complex requirements have detailed discussion beforehand
         iv. Topic pushed to next CoP
4. Topic Collection
   a. Large or complex Requirement Splitting
   b. Estimation of user stories and actual effort
   c. Conflict in Roadmaps (how to deal with conflict in plans)
   d. Motivating developers
5. Next Moderator
   a. D[redacted]
   b. Date 26.03
6. We'll have a short Feedback session (10 mins)

Figure 4.29.: Screenshot of instantiated Community of Practice for Product Owners - Documentation contd.

In the End of the CoP we used the last minutes to hold a short feedback session, the results are presented in Section4.5.7. Finally, we discussed the method, the newly instantiated CoP should follow to evaluate their CoP, see Figure4.30.

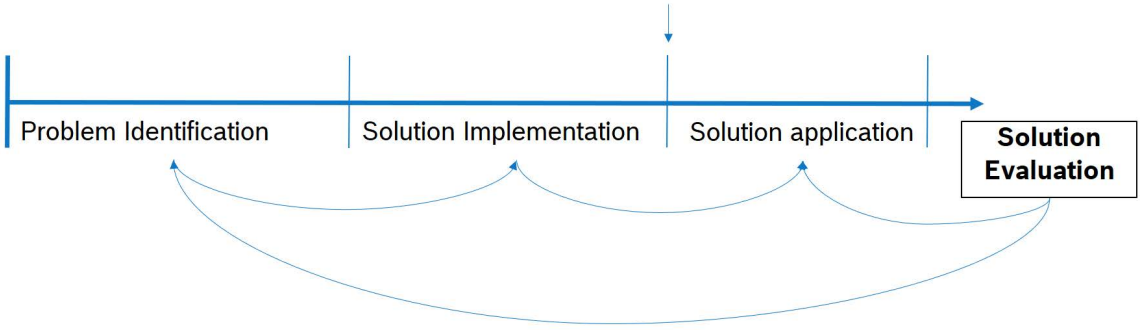


Figure 4.30.: Process proposed for Evaluating instantiated Community of Practice at Technology LLC.

Making sure to evaluate the solution after some applications and revisit the Problem Identification and Solution Implementation and Application and if necessary changing those instances as to improve the solution and make it more fitting to the LSADP of Technology LLC. Overall the adoption of the CoP fulfilled seven of the characteristics of a successful CoP and only limited the characteristic of an open community for the first phase of the CoP.

4.5.5. Deviations configured Design and instantiated Solution

This section aims at summarizing the deviations between the pattern introduced and the instantiated solution at Technology LLC. for **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution*. Therefore, we will focus on each difference in application. As far as the general **Pattern Overview**, **Example**, **Context**, **Variants**, **Consequences** and **See Also** are concerned of the LSADPL there are no deviations between the configured Design and instantiated Solution. Deviations can be observed in the specifications of the **Problem**, **Forces** and **Solution** building blocks of the LSADPL. The final instantiated pattern is provided after discussing the deviations.

Problem

The Problem spectrum addressed by the instantiated pattern is a little larger, as the recurring concerns **C-81***How to understand all interfaces and dependencies of the system?* and **C-80** *How to manage overarching backlog item prioritization with multiple product owners?* are added to the list of problems addressed by the instantiated Coordination-Pattern **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution*. These recurring concerns where identified in the LSADP of Technology LLC. and are all connected to to the Product Owners concerns.

Forces

In Comparison to the configured design, there is one new Force identified as to why the LSADP hasn't had knowledge share across the POs. The PO of the LSADP of Technology LLC. are already stretched thin through having to have active contact with multiple key users and customers and have to coordinate their activity in multiple agile teams. Resulting in: 'Product Owner are involved in multiple agile teams and have to keep contact with several key users and do not see another meeting as solution as they already face multiple challenges and feel they do not have enough time to address them. Limiting their possibility to openly discuss their practices applied and not theories.'(compare **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution*) being added to the Forces segment.

Solution

As the **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated solution* wants to use the CoP to additionally share module knowledge they decided to limit the participants to only POs from the LSADP. Creating a concept more accurately comparable to the chapter concept of Spotify, as only one role is represented of one project team. This is represented by the solution section not including 'the openness to anybody interested'[40] statement. Additionally the **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution* is lead by a group of leaders, therefore adding this to the Solution.

4.5.6. Community of Practice for Product Owners - Instantiated Solution **

Since it is a new application of CoP the maturity level for the newly added problems has to be controlled.

CO-Pattern Overview	
ID	CO-1-Instantiated
Name	COMMUNITY OF PRACTICE
Alias	-
Summary	To facilitate knowledge sharing, Communities of Practice can be setup. 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 Technology LLC., the Agile Coach saw a need for inter-team knowledge share, as different methods were used in the teams.

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

Problem
Following concerns are addressed by this pattern:

- **C-05** *How to facilitate shared context and knowledge?*
- **C-19** *How to deal with Internal Silos?*
- **C-39** *How to establish a Culture of Continuous Improvement?*
- **C-80** *How to manage overarching backlog item prioritization with multiple product owners?*
- **C-81***How to understand all interfaces and dependencies of the system?*

Forces
Following forces have been identified:

- Product Owners are involved in multiple agile teams and have to keep contact with several key users and do not see another meeting as solution for them.As they already face multiple challenges and feel they do not have enough time to address them all. Limiting their possibility to openly discuss their practices applied and not theories.
- Facilitating shared context and knowledge across the organization is difficult

- Internal silos create gaps in knowledge and communication between agile teams

Solution
Community of Practice for Product Owners:
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'[64] . Participation is for now limited to POs from the same LSADP of Technology LLC. as the participants decided to use it to share module knowledge as well. The CoP is comparably open to the Chapters of the Spotify model. The intention is to enable frequent knowledge and expertise sharing between the participant[64] . 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 [64] . 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[40]. In addition, each Community of Practice should be lead by an expert or a group of experts, who are passionate to make the event a success and keep it on a frequent level[40]. 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 [40].

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
Following Benefits have been associated with this pattern:

- Encouraging knowledge sharing for diverse topics
- Breaking up silos
- Enabling a culture of continuous improvement

Following Liabilities have been associated with this pattern:

- Requiring an active involvement of participants
- Topics in the agenda could be too diverse and broad
- Providing right incentives to the participants is challenging

Known Uses

- Technology LLC.
- Electronic GmbH
- Global Insurance Corp
- LuxCarsCorp
- Retail Corp
- Software Inc.

4.5.7. Lessons Learned from instantiated Community of Practice for Product Owners at Technology LLC.

After the first instantiated CoP for POs was finished, we made a short feedback session using the QuestionnaireA.3. Some key observations during the feedback session and the process of the CoP was that the overall acceptance of the Pattern together with the characteristics to make it successful was high.

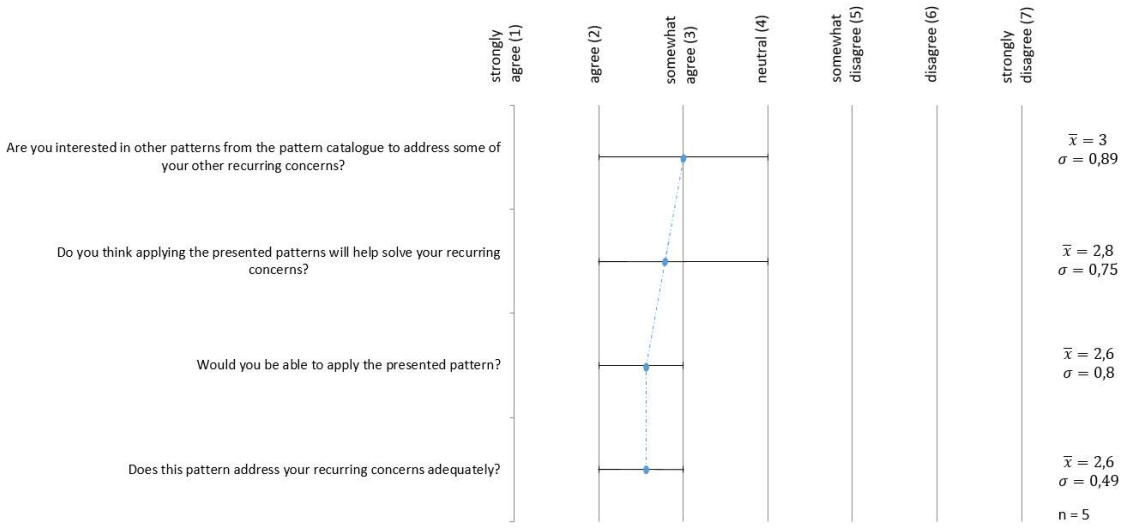


Figure 4.31.: Evaluation of instantiated Community of Practice for Product Owners

For the instantiated **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution* we had four of the six interviewed POs attending and one PO who attended the second presentation date. In total five POs build the first implementation of the CoP. PO-1, PO-2,PO-3,PO-5 and PO-7 (joined in after presentation of solution designs) then answered the feedback questionnaire. The results, (see Figure4.31) indicate, that the **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution* was sufficiently explained and addressed the concerns of the POs.
Additionally, they agreed (with an average of 2,6) that they could have applied the **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution* on their own. However, they only somewhat agree, that they would be interested in applying other patterns of the presented solution designs of the pattern catalog[61]. All feedback indicated that they will be continuing the application of the **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution* on their own.
'We had a good start and need to see how it develops from now on.' Product Owner (PO-7)(2020)
Additionally, all POs would recommend the **CO-1** *Community of Practice* to their colleagues with one PO stating:
'Yes definitely! It' always beneficial to have a KT.'Product Owner (PO-3)(2020)

Regarding what could be improved about the **CO-1** *Community of Practice* one PO mentioned the *'Awareness and Willingness of POs' Product Owner (PO-1)(2020)* needs to be addressed. Overall, the willingness to apply this and the positive feedback in regards to the **CO-1** *Community of Practice* pattern indicate that the pattern language in this instance lead to a good documented pattern in the pattern catalog. This indicates that the patterns of the pattern catalog can achieve their purpose.

5. Discussion

In this chapter we will revisit the Key Findings of the Case Study and how they answer the Research Questions formulated in this thesis. We will compare our key findings to the related work and see where they add or challenge existing findings from the literature. We will finish with a look at the Limitations of our Key Findings.

5.1. Key Findings

This Thesis uncovered several interesting findings of the adoption, concerns, pattern candidates and the learning process of a LSADP through a case study in the technology sector. These findings answer the research questions stated for this thesis.
RQ1. How has LeSS been adopted and applied at the case study partner?
Through the conception of the agile process4.4 based on observations and three structured experts interviews, we identified the Adoption of two basic LSADP frameworks with LeSS and Spotify to create a tailored LSADP framework for Technology LLC.
The learning observed from Technology LLC. identified during the expert interviews and observations was that the mistake of adopting LeSS one to one in the beginning led to a free process of picking fitting practices from different frameworks to generate an individual LSADP framework. While Technology LLC. focused on creating their individual LSADP they introduced new concepts of existing agile practices.
One major Finding in regards to the first RQ is the use of purpose teams, with domain expertise and therefore the creation of domain-specific feature teams similar to the findings of Paasivaara et al.[39], who identified cross-site and specialized teams. Overall, the LSADP went through two phases of agile transformation so far. Similar to the observed phases in the works of Fuchs and Hess[25] and Paasivaara et al.[39], the LSADP agile phases were initiated by a push to become more agile.
As this case study ended, there were first indicators of another agile transformation phase being initiated, as the LSADP size was increased combining two LSADP into one larger LSADP.
The purpose teams have another unique trait, as the PO are assigned to modules and through their module assignment, become part of the purpose teams which are responsible for developing these modules. This makes POs part of one or many agile teams. In addition to the purpose teams, the LSADP addressed their issue of working on a platform with high degrees of domain-expertise, which was the introduction of a new role. The Process Consultant is an admitted knowledge bottleneck, who through his special role has

half of his working time, reserved for supporting other stakeholder groups of their team.
Overall, the LSADP went through two phases of agile transformation similar to the observed phases in the works of Fuchs and Hess[25] and Paasivaara et al.[39]. Agile phases initiated by organizational or inter-team pushes to become more agile.

RQ2. What are recurring concerns of stakeholders at the product organization of the case study partner?

Regarding the recurring concerns, we identified a total of nine new recurring concerns through the semi-structured interviews with the stakeholder groups DEV, PO and SM.
Additionally, we could identify and verify a total of 56 already observed recurring concerns by Uludağ et al.[57] for these three stakeholder groups, however, not all of these identified concerns were typically assigned to the stakeholder groups. We identified that the SM ofLSADP of Technology LLC. shared many concerns originally assigned to agile coaches, which originates in their special part-time SM role.
Additionally, many concerns related to the Test Team were identified by the DEV as they are responsible for establishing automated testing and are supported by manual testers from an external organization. By creating their individual LSADP practice, most of the newly identified concerns are dealing with multiple POs in a team and sharing knowledge across those domain-specific teams.
For the DEV stakeholders we identified their main concern category to be Quality Assurance. For the POs the main concern category was Requirements Engineering, while many of the identified Requirements Engineering concerns had their root-cause in Knowledge Management. Finally, the SM stakeholders were concerned with Culture & Mindset problems.

RQ3. What are good practices for addressing recurring concerns of stakeholders of the product organization of the case study partner?

In Combination with the recurring concerns, we were able to document a total of 35 good practices (18 Methodology, 11 Coordination and seven Viewpoint) and three principle candidates. Additionally, discovering one Coordination-Pattern **CO-1** *Community of Practice*, also documented by Uludağ et al.[55], which through the combination of LeSS and Spotify serve an important role of creating a Knowledge Share platform across the agile teams of Technology LLC, as described by Wenger et al.[64].
Overall the pattern candidates indicated a intense focus on addressing the domain-specific expertise concerns, which Technology LLC. created to address issues at the initial agile transformation phase. With the goal of creating feature teams in the future, the LSADP created multiple practices to foster the knowledge share of their LSADP, especially by introducing domain experts with a 50% capability of addressing knowledge share with their team colleagues.

RQ4. Which bad practices should be avoided in the product organization of the case study partner?

This Thesis documented 11 anti-pattern candidates from the interviews with the stakeholder groups. In total five Anti-Patterns were connected to Knowledge Management concerns. Additionally we identified one anti-pattern addressing once more the initial

agile transformation phase of the LSADP, the anti-pattern **A-1** *Don't use Frameworks as Recipes* was also discovered by Uludağ et al.[59]. Overall, the high number of anti-pattern candidates documented indicates that misusing agile practices or misunderstanding agile values is a major challenge for LSADP, one that if not aware of creates a chance of becoming 'pseudo agile'[59].
When combining the findings from RQ3 and RQ4 we identified three concern categories, which were relevant for the final Research Question, where this thesis implemented a pattern from the pattern catalog at Technology LLC.
RQ5. What are the lessons learned of implementing already observed best practices in the product organization of the case study partner?
The concept of the proposed pattern allowed for an easy guided implementation of the **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated Solution*. The content of the pattern was applicable and the new formed CoP agree the pattern addresses their concerns and will help in addressing them in the future. Overall, the pattern language presented only little deviations between the design and instantiated solution. With a consensus, that the participants of the **CO-1-Instantiated** *Community of Practice for Product Owners - instantiated solution* agree the current template for documenting patterns makes them applicable without guidance needed, see Section4.5.7.

5.2. Limitations

Limitations of this thesis follow the identified threats to validity by Runeson and Höst[45].
Construct validity -*To what extent does the studied object reflect what the researcher is focused on[45].*
By gathering data through observations,interviews with different stakeholder groups and attending several meetings of the LSADP studied, the data collection followed a mixed-methods approach.
Internal validity - *Causal relations are misinterpreted by the researcher, ignoring forces out of scope[45].*
The first part of this work was a descriptive case study, hence no threat of the internal validity being challenged. The second part documented relationships between concerns and best practices applied, to not unintentionally ignore forces on these relationships, we cross-checked the findings with several interview partners, to gain as big a picture as possible. The time-horizon of the deviations from the configured design and the observed instantiated solution are challenged by internal validity. As the results of the Section4.5.7lessons learned are based on a single occurrence of the implementation. While they offer insights into the action plan and can create a degree of validity for the relationship of Context, Problem and Solution. There might be some forces ignored, as we weren't able to observe the process and evolution the **CO-1-Instantiated** through further being applied.
External validity - *Generalization and relevance for other cases[45].*

By applying the LSADPL by Uludağ et al.[55], this thesis aimed at using a construct of documenting the findings, which allows for application at other organizations. Due to the fact, the case study partners LSADP has been tailored to their fit, there exists the chance for some best practices observed being unique context, problem and solution for this case.
Reliability - *Dependency on Researcher[45].*
While the data collection and analysis was mainly done by one researcher, the application of the how the data was collected and which information was gathered followed the combined work of the larger research project. Further, one audio recording of an interview corrupted and we had to go through all documented findings of that interview in detail with the interviewee to make sure, the data was correct. For example, the structured-questionnaires was only minimally adapted to gather important information for the observed case study partners' LSADP. Additionally the application of the LSADPL allows for a general measure for documenting best practices, which is understandable and applied by several researchers. Therefore, the here documented practices would be documented in the same fashion by different researchers.

6. Conclusion

This chapter summarizes the thesis and research results and offers this researchers opinion on what future work would be interesting building on this work.

6.1. Summary

The motivation for this thesis was the ongoing application of agile practices in a large-scale context. We first aimed at documenting and creating a individual representation of the case study partners LSADP.
By applying the Pattern-based Design Research, we aimed at documenting and identifying concerns and best practices observed at the case study partner as well as introducing patterns from other organizations to resolve some concerns of the case study partner.
Following these approaches this thesis created a LSADP of Technology LLC. by conducting expert interviews and using observations. Observing the agile transformation phases of Technology LLC. and documenting the current LSADP process. Then we conducted 11 more interviews with three different stakeholder groups at Technology LLC., identifying 56 concerns from the literature and nine new concerns. These concerns stemmed from the adoptions done at Technology LLC. and their resulting unique LSADP with new team concept and new roles.
Due to the case study being conducted at a single organization, this thesis documented 36 good and 11 bad practices, as well as three principles, which found application at Technology LLC. Together with the one pattern and one anti-pattern observed at Technology LLC. , the overall approach of solving the concerns of the stakeholder groups can mainly be boiled down to the implementation of the new roles. Additionally, we could verify one Coordination pattern through applying it to address concerns of Technology LLCs' PO stakeholder group. By verifying the pattern, we could also provide some verification for the LSADPL used in this thesis.

6.2. Future Work

As this thesis documented several new pattern candidates, it would be interesting to see, whether these can be found at other organizations or even whether they could be applied to other organizations, for which further case studies of Technology LLC. and other organizations would be useful.
Further, a case study following up this work would be interesting, as the lessons learned

of the introduced patterns could be observed and the agile transformation phase currently happening could be documented. Additionally, a follow-up case study would provide insights whether the introduced and instantiated patterns are applied and which deviations exist to the documented patterns.
Finally, further work needs to be done in implementing the patterns of the pattern catalog at multiple organizations. Not only to further develop the LSADPL, but also offer more validity, by observing the deviations when applying the patterns. In total, more research, applying the full Pattern-based Design Research approach, would help validate the LSADPL and the patterns documented in the pattern catalog. Therefore, we also recommend further research in applying the full Pattern-based Design Research approach at other organizations.

A. Appendix

A.1. Interview Questionnaire for Identifying the adoptions of the Agile Program at the case study partner

General Information
Questions about the Participant

- What is your role description in your organization?
- How long have you been active in your role in the field of scaled agile software development?

1. **Introduction of LeSS**
   a) When was LeSS introduced and how long did the Implementation take, respectively how long has the implementation been going on?
   b) Why and focusing on which criteria has LeSS been implemented and which existing concerns was the framework supposed to address?
   c) Why did you choose LeSS?
   d) With which agile frameworks are you combining LeSS?
   e) What training did employees have concerning LeSS?
   f) Which challenges and problems did you face when implementing LeSS? Were several adaptions necessary to integrate LeSS in your agile program?
   g) What are your Lessons Learned concerning the implementation of LeSS?

2. **Adaption and Application of LeSS**
   a) **General**
      i. Who are stakeholders in your LeSS?
      ii. How has LeSS been embedded in the organization structure?
      iii. Are you using LeSS or LeSS Huge?

b) **Goals**

i. How are you applying which LeSS principles?

ii. How and to which degree is coordination (team intern, team extern, to other stakeholders) regulated?

iii. Which of the LeSS guidelines regarding principals, coordination and communication do you deem most useful and where would you see a general need for improvements?

c) **Roles**

i. Which roles are represented in your agile program?

ii. What classifies work, responsibilities and area of duty for the respective role (Product Owner, Scrum Master, and Development Team)?

iii. Where do you see a need for improvement concerning roles and their responsibilities and area of duty?

d) **Artifacts**

i. What artifacts are represented in your agile program?

ii. What denotes the content of these artifacts (Sprint Backlog, Sprint Goal, Product Increment and Definition of Done; i.e. are there several persons responsible or only one)?

iii. How is the Product Backlog managed and how does the assignment from the Product Backlog to a team work?

iv. Is each artifact defined and meaningful? Where do you see a need for improvement?

e) **Processes**

i. Which processes are part of your agile program?

ii. What generally denotes your processes (Sprint, Product Backlog Refinement, Sprint Planning 1 and 2, Daily Scrum, Sprint Review, Overall/Team Sprint Retrospective, i.e. Goals, Procedure, and Participants)?

iii. How meaningful do you deem your processes and where do you think is a need for improvement?

f) **Architecture**

i. Which role do architects (i.e. IT architects, EA architects) assume in your agile program? How do you conceive your cooperation?

ii. What influence has architecture in your development process?

iii. What is your opinion regarding CoP in architecture topics? Are you participating? Do you deem it helpful?

iv. Which tools or models do you use for visualization of the architecture?

v. How do you deal with technical debt?

vi. Where do you see a need for improvement concerning architecture topics?

3. **General Retrospective of LeSS Framework**

   a) How satisfied are you with the framework (Scale 1-5, 1 = very unsatisfied, 5 = very satisfied)? Please explain your answer.

   b) Does LeSS live up to your expectations, where are possibilities for improvement and where were you disappointed with the framework?

4. **Discussion**

   a) Would you be willing, in the context of this case study, to take part in a final questionnaire? If yes, please provide your name and e-mail.

   b) Remarks and Questions

A.2. Semi-structured Interview Questionnaire for Identifying Concerns and Documenting Good and Bad Practices

General Information

1. **General Information about the Participant**
   a) What is your role called at your company?
   b) Which role description applies to you?
   c) How many years have you been active in the field of scaled agile software development?
   d) May we contact you again in the context of the study if necessary?

2. **Identification and Description of Concerns and Practices**
   a) What are the recurring concerns you face in your role?
   b) On what level do these typically occur and how often are you confronted with them?
   c) In which category would you classify the concern?
   d) Which practices do you apply to address the concern?
   e) Which practices should not be applied to address the concern?

3. **Identification of Recurring Concerns and Description of Practices**
   a) Which of the following concern did you face in your role?
   b) At what level do they typically occur and how often are you confronted with them?
   c) In which category would you classify the concern?
   d) Which practices do you apply to address the concern?
   e) Which practices should not be applied to address the concerns?

4. **Discussion**
   a) Do you have any comments or open points?

A.3. Questionnaire: Pattern Feedback

General Information

1. **Questions about the Participant**
   a) Which role description applies to you?
   b) How many years have you been active in the field of scaled agile software development?

2. **General Retrospective introduced patterns**
   Please answer below on a scale from 1-5 (1 = disagree, 2 = somewhat disagree, 3 = neutral, 4 = somewhat agree, 5 = agree)
   a) Do the presented patterns address your recurring concerns adequately?
   b) Would you be able to apply the presented patterns?
   c) Do you think applying the presented patterns will help solve your recurring concerns?
   d) Are you interested in other patterns from the pattern catalogue to address some of your other recurring concerns?

3. **Feedback introduced Pattern**
   a) How do the introduced patterns address your recurring concerns?
   b) Which of the introduced patterns will you apply and how? Will you continue to apply the introduced pattern?
   c) Would you recommend the applied pattern to your colleagues? Where do you see a need for improvement in the applied pattern?

B. Appendix

B.1. Documentation of newly identified Concerns

The following tables consist of all newly identified concerns, which were identified through the interview process.. The concerns mentioned are categorized according to the LSADPL by Uludağ et al.[55]. Hence, id, name, description, category, scaling level, and the source are documented.

ID	Name	Description	Category	Scaling Level	Source
C-79	*How to balance amount and quality of delivered requirements?*	Even though as a Product Owner, one's main goal is to further deliver business value to customers, the balance of quality and delivery has to be kept. But how do you make customers' aware of the benefits of delivering less, but of higher quality?	Quality Assurance	Team	PO-1
C-80	*How to manage overarching backlog item prioritization with multiple product owners?*	With a Scaled-Agile Development Program consisting of at least Product Owners per team, the POs have to coordinate with each other which backlog items make the cut to be added to the Sprint Backlog. Prioritization is difficult, and conflict will ensue, if there is no clear method for dealing with this concern.	Communication & Coordination	Program	PO-1
C-81	*How to understand all interfaces and dependencies of the system?*	Interviewee PO-5 described the issue of understanding all dependencies between the large amount of modules in the system. When creating user stories and requirements, it becomes difficult to then make sure the whole system is considered with a limit view of the system. This is a concern for the Enterprise Architect.	Knowledge Management	Program	PO-5

ID	Name	Description	Category	Scaling Level	Source
C-82	*How to support an On-Boarding approach for different stakeholder groups?*	This is a concern for Scrum Masters, with a growing project and PO being part of the agile team, the On-Boarding material has to be reflecting the different viewpoints of all stakeholders accurately. Otherwise On-Boarding is an issue.	Knowledge Management	Program	PO-2
C-83	*How to manage requirement development for multiple teams?*	In case of PO-6, who is part of multiple agile teams at once, it is an issue to manage development at these different teams and continue having regular check-ins with customers. Keeping track of all developments becomes more difficult, as multiple modules can be concerned and interaction with multiple external partners might be required.	Project Management	Team	PO-06
C-84	*How to involve all team members in solution generation?*	Scrum Master SM-1 identified an issue in his team, the introduction of Process Consultants, while useful for many reasons, limited the engagement of other developers. When only one developer actively works on solution generation and all other team members accept his solution, the solutions are limited to one mind, instead of using the knowledge of a whole team.	Culture & Mindset	Team	SM-1

ID	Name	Description	Category	Scaling Level	Source
C-85	*How to share domain knowledge across agile teams?*	DEV-1 identified, that he as a developer has issues to gather knowledge from other agile teams, when not having to deal with something that stops him from working. The sharing of domain knowledge only occurs in teams of that domain and developers interested in the domain knowledge have a higher boundary to receive that knowledge. DEV-1 himself had issues sharing his knowledge with other agile teams in a efficient manner.	Knowledge Management	Program	DEV-1
C-86	*How to involve remotely working and external colleagues?*	Dev-2 mentioned that he as a external colleague has issues connecting to the video-chats without technical issues, when not working from the SADP offices. This is a concern for the Scrum Masters.	Tooling	Organization	DEV-2
C-87	*How to clarify details outside of meetings in cross-shore agile teams?*	DEV-5 had the recurring concern to get a hold of team members not co-located, which can especially painful when trying to gather some explanation to a question, which in turn makes the developer have to change tasks until he receives the answer. While usually communication is direct, sometimes due to the time difference, developers still have to wait a few hours to clarify an issue.	Communication & Coordination	Team	DEV-5

B.2. Documentation of existing identified Concerns

- **C-03** :*How to coordinate geographically distributed agile teams?[57]*
- **C-04** :*How to deal with doubts in people about changes?[57]*
- **C-05** :*How to facilitate shared context and knowledge?[57]*
- **C-07** :*How to deal with incorrect agile practices?[57]*
- **C-10** :*How to create precise requirement specifications for the development team?[57]*
- **C-12** :*How to provide sufficient tools and infrastructure for remote communications?[57]*
- **C-13** :*How to share common vision?[57]*
- **C-15** :*How to elicit and refine requirements of end users?[57]*
- **C-16** :*How to deal with increasing workload of key stakeholders?[57]*
- **C-18** :*How to split large and complex requirements into smaller requirements?[57]*
- **C-19** :*How to deal with internal silos?[57]*
- **C-20** :*How to facilitate communication between agile teams and other teams using traditional practices?[57]*
- **C-22** :*How to balance short-term and long-term goals?[57]*
- **C-23** :*How to establish a common scope for different stakeholder groups?[57]*
- **C-24** :*How to create team spirit and trust amoung agile teams?[57]*
- **C-26** :*How to align and communicate architectural decisions?[57]*
- **C-28** :*How to communicate business requirements to development teams?[57]*
- **C-29** :*How to facilitate agile teams to participate at cross-shore meetings?[57]*
- **C-30** :*How to synchronize working hours of cross-shore meetings?[57]*
- **C-31** :*How to deal with geographical distance between agile teams?[57]*
- **C-32** :*How to deal with lacking team cohesion at different locations?[57]*
- **C-33** :*How to build trust of stakeholders in agile practices?[57]*
- **C-35** :*How to define clear and visible priorities?[57]*
- **C-36** :*How to establish automated testing?[57]*

- **C-37** :*How to create lightweight documentation?[57]*
- **C-39** :*How to create a culture of continuous improvement?[57]*
- **C-41** :*How to deal with unplanned requirements and risks?[57]*
- **C-42** :*How to rearrange physical spaces?[57]*
- **C-43** :*How to enforce customer involvement?[57]*
- **C-44** :*How to deal with communication gaps with stakeholders?[57]*
- **C-45** :*How to deal with black and white mindsets?[57]*
- **C-46** :*How to deal with closed mindedness?[57]*
- **C-47** :*How to deal with higher-level management interferences?[57]*
- **C-49** :*How to deal with increased efforts by establishing inter-team communication?[57]*
- **C-50** :*How to deal with lacking sense of ownership responsibilities for developed services?[57]*
- **C-53** :*How to ensure traceability of tests and requirements?[57]*
- **C-54** :*How to make a cost and schedule estimation?[57]*
- **C-55** :*How to create a teamwork centric rewarding model?[57]*
- **C-56** :*How to define clear roles and responsibilities?[57]*
- **C-59** :*How to establish a common understanding of agile software development?[57]*
- **C-60** :*How to create and estimate user stories?[57]*
- **C-61** :*How to deal with cultural differences between cross-shore agile teams?[57]*
- **C-62** :*How to deal with fixed price contracts in agile software development?[57]*
- **C-63** :*How to explain requirements to stakeholders?[57]*
- **C-65** :*How to deal with office politics?[57]*
- **C-67** :*How to encourage development teams to talk about tasks and impediments?[57]*
- **C-68** :*How to write understandable automated tests?[57]*
- **C-69** :*How to establish requirements verification?[57]*
- **C-70** :*How to define high-level requirements a.k.a. epics?[57]*
- **C-71** :*How to measure the success of the large-scale agile development program?[57]*

- **C-72** :*How to consider required competencies when assigning teams to tasks?[57]*
- **C-73** :*How to deal with decreased predictability?[57]*
- **C-74** :*How to empower agile teams to make decisions?[57]*
- **C-75** :*How to form and manage autonomous teams?[57]*
- **C-76** :*How to coordinate test and deployment with external parties?[57]*
- **C-77** :*How to build an effective coaching model?[57]*
- **C-78** :*How to synchronize sprints in the large-scale agile development program?[57]*

C. Appendix

The Practices are documented with the maturity level as defined by Uludağ et al.[55]. The maturity of a pattern is represented by star notation: two stars, indicating the pattern addresses a genuine problem, one star indicates the pattern addresses a real problem, no star means the pattern was useful for a observed problem but needs revision[55].

C.1. Documentation of Coordination Pattern and Good Coordination Practices

The Pattern **CO-1***Community of Practice* was already presented in Section4.4.1.
The good coordination practices **CO-04** *Kickoff* and **CO-08** *Newsflash* were already presented in the findings in Section4.4.4.

C.1.1. Product Backlog Refinement**

CO-Pattern Overview **	
ID	CO-05
Name	PRODUCT BACKLOG REFINEMENT
Alias	-
Summary	Product Backlog Refinement for estimating backlog items and clarification of these Backlog items. Constant communication between PO and team to have a good understanding of requirements

Example
Technology LLC. uses the PBR to elicit backlog items and create a common understanding in the team. As requirements are prone to change, the PBR is used to reiterate estimates and explain requirements from the Backlog.

Context
Backlog items need to be estimated and be commonly understood by all team members so everyone could be able to solve them and knows what is required.

Problem
The following concerns are addressed by this practice:

- **C-60** *How to create and estimate user stories?*

Forces
Following forces have been identified:

- Estimates can vary depending on experience of developers, therefore accurate estimates have to be aligned.
- Common understanding of a task and requirements required, so the functionality represents the business need.

Solution
Product Backlog Refinement: PO presents requirements followed by short discussion and estimation. As the event is repeated twice a Sprint, discussions can be lengthy if required, but don't have to be lengthy. Focus is to get a common understanding of all backlog items, so when they are to be implemented, everybody knows what to do. Afterwards estimation on Story Points via **M-17** *Planning Poker Light*

Consequences
The following Benefits have been associated with this practice:

- Constant knowledge share.
- Clearer and more accurate estimations.

The following Liabilities have been associated with this practice:

- Time expensive with one hour every week. –> A quarter of parallel woking hours.

See Also
The practice uses **M-17** *Planning Poker Light* for estimating user stories.

C.1.2. Pre-Planning Coordination

CO-Pattern Overview	
ID	CO-01
Name	PRE-PLANNING COORDINATION
Alias	-
Summary	To avoid conflicts in Sprint Planning and to organize a more efficient Sprint Planning, the Product Owners coordinate their topics before the Sprint Planning without involving the Team.

Example
The agile program of Technology LLC. has multiple customers and modules supported. Each module has a Product Owner (PO), each team works on multiple modules according to their domain knowledge. Multiple PO are working in one team, to coordinate backlog items resolved in a sprint, each team has a head PO and there is a CPO. However, to keep a healthy balance a Pre-Planning Coordination occurs to manage overarching backlog item prioritization.

Context
Multiple PO in one team with own interest of improving supported modules. Coordination of backlog item prioritization needs to happen.

Problem
Following concern is addressed by this practice:

- **C-80**:*How to manage overarching backlog item prioritization with multiple product owners?*

Forces
Following forces have been identified:

- Multitude of modules supported in system used by multiple customers.
- Each PO aims at improving the modules he is responsible for, with multiple POs in a single agile team, they have to coordinate.

Solution
Pre-Planning Coordination:
Head PO of each team organizes a short meeting or asks each PO individually about their most important requirements for the following Sprint. Overall, the **V-06** *Epic Plan Game Board* dictates the highest priority for the next sprint. Limiting the cause for conflict as the epic plan has priority and should always be followed. As all PO know the overall capacity of the team and the amount of capacity needed for their highest priority requirements, the

discussion usually is very short. When there is a conflict, which cannot be resolved by the conflict parties, the head PO has final say about which requirements' priority is higher. This coordination happens before every team meeting, which is connected to discussion about backlog items.

Consequences
The following Benefits have been associated with this practice:

- Common understanding in PO Team for most important topic in the following sprint.
- More efficient follow-up meetings such as sprint planning, PBR.

The following Liabilities have been associated with this practice:

- Development team is not part of the process.
- Decisions from Head PO can negatively influence morale in PO team.

See Also
This practice is applied before **CO-05** *Product Backlog Refinement* and has an direct connection to the **M-11** *Product Owner Team*.

C.1.3. Face-to-Face Knowledge Transfer

CO-Pattern Overview	
ID	CO-02
Name	FACE-TO-FACE KNOWLEDGE TRANSFER
Alias	-
Summary	Scheduled meetings, when impediments has been raised. Process Consultant and respective colleague discuss impediment and share Knowledge.

Example
Technology LLC. uses Process Consultants with expert knowledge in a specific domain. When a requirement assigned to a developer with less experience or when discussing impediments in a daily, the Process Consultant proactively offers up F2F meetings to resolve these impediments.

Context
Complex System with migrated legacy systems, requiring in-depth domain knowledge. On-Boarding of new employees and Impediment discovery during implementation.

Problem
Following concerns are addressed by this practice:

- **C-18** :*How to split large and complex requirements into smaller requirements?*
- **C-19** :*How to deal with Internal Silos?*
- **C-84** :*How to involve all team members in solution generation?*
- **C-85** *How to share domain knowledge across agile teams?*

Forces
Following forces have been identified:

- Legacy Code with little to none technical documentation within a complex and vast system. Assumption developers understand specific platform part or process can hinder a sufficient explanation.
- Domain knowledge difficult to share among all members, as well as different experience levels in developers.

Solution
Face-to-Face Knowledge Transfer:
The following application scenarios and practices exist:

1. On-Boarding of new employee in a team
   a) PC introduces employee via F2F (remote or in-person) to domain and explains most important functionality and architectural setup to new employee.
   b) PC shows graphically within system, first from customer view then in the code base.
   c) Creating a contact point for further questions.
2. Impediment Discovery in Daily or during Implementation
   a) PC blocks a time block after daily (i.e. 30 min) for explanation or discussion.
   b) PC proactively invites affected developer or developer asks for PC's help during daily or beforehand.
   c) F2F meeting (remote or in person) explaining requirement or impediment root from GUI to code base.
   d) Check PC and Developer whether Impact Analysis is required.
   e) Assuming new employee with no knowledge, so visualization through GUI explains existing implementation best.
3. Product Owner requires support of Process Consultant
   a) Product Owner asks for support.
   b) Meeting set to discuss impediment of Product Owner (i.e. estimation of Requirement, need for split).
   c) PC estimates and explains needed changes eventual Proof of Concept required.
   d) Next PBR discussion in whole team presented by PO.

Consequences
The following Benefits have been associated with this practice:

- Saving time of team by having separate meeting for explanations.
- Supported Knowledge Transfer with visual help as well as expert knowledge share.
- Easier process for whole team.

The following Liabilities have been associated with this practice:

- Limits experts' development capabilities.
- When expert is not available, the Knowledge Transfer eventually has to wait

See Also
Initiated by the the expert**M-07** *Process Consultant* and eventually connected with a **M-18** *Proof of Concept*.

C.1.4. Exemplary Knowledge Transfer

CO-Pattern Overview	
ID	CO-03
Name	EXEMPLARY KNOWLEDGE TRANSFER
Alias	-
Summary	Scrum Master, who is Developer as well, actively instigates knowledge transfer sessions with developers to help them become more confident and showcase good ideas.

Example
Team A of a large-scale development program has a requirement creating a necessity of an architecture decision. The assigned developer and the SM open up a meeting for discussing the solution options, as the required implementation offers a good example or is difficult to solve.

Context
Requirement or solution finding offers a teachable moment about implementation possibilities, which is beneficial for whole team.

Problem
Following concerns are addressed by this practice:

- **C-33** *How to build trust of stakeholders in agile practices?*
- **C-39** *How to create a culture of continuous improvement?*

Forces
The following forces have been identified:

- For people to share good solutions or practices and be put in the spotlight can be uncomfortable.
- Geographically distributed teams with limited amount of shared working hours.

Solution
Exemplary Knowledge Transfer:
After Requirement estimation in PBR or when a Developer contacts SM about Impediment. Spontaneous team meeting, either in blocked time right after daily or whenever possible, but as close as possible to discovery. If not possible to meet right away, it is pushed to the next retrospective. Goal of meeting is to communicate and discuss a major implementation issue with possible architectural implications. SM organizes and opens the meeting, making sure to involve possibly required experts. Developer explains the issue he faces.

Team Discussion on possible solution.

Consequences
The following Benefits have been associated with this practice:

- Knowledge Transfer in Team. Solution generation via team decision.
- Focus on architectural discussion encourages culture of continuous improvement.
- Common understanding of solution created and eventually related architectural concept.

The following Liabilities have been associated with this practice:

- Difficult to organize. Taking away time of multiple people.
- Useless when no discussion is generated, but rather everybody just listens

See Also
This practice is used on combination with the discovery in **CO-05** *Product Backlog Refinement* or Daily.

C.1.5. Direct Customer Communication

CO-Pattern Overview	
ID	CO-06
Name	DIRECT CUSTOMER COMMUNICATION
Alias	-
Summary	Product Owner use direct customer communication to involve customers more in the Large-Scale Agile Development Program and to create a higher customer satisfaction. They don't wait for customer to ask for something, rather going towards them and present something.

Example
Technology LLC. has a project team working in a large-scale agile development program. The customers of the team have requirement and if they are key-users, they have an increased involvement in testing the delivered features as an acceptance test.

Context
Customers with little interest in being involved in the process of LSADP or working in traditional processes, hence, not aware the amount of required involvement and impact the involvement has on the quality of the product.

Problem
Following concerns are addressed by this practice:

- **C-28** :*How to communicate business requirements to development teams?*
- **C-43** :*How to enforce Customer Involvement?*

Forces
Following forces have been identified:

- Customers are not used to being part of the process outside of giving requests and using the final product.
- Customers want to be informed about progress of their functionalities, especially in connection with delays or issues related to the functionality.
- Customers might feel that the delivered product doesn't represent their business need.

Solution
Direct Customer Communication:
Focusing on direct communication between PO and customer, creating a personal relationship and proactively requesting testing from key users and customer. Additionally asking for a formal approval of the delivered requirement via a soft go-live. Including customer as an informal manual tester. Actively involving customers in team events, when the situation allows for it: Involving the customer in a **CO-05** *Product Backlog Refinement* and Review when closely working on a complex requirement. Having the customer directly explains to the team how he uses the system and why certain changes are important for him.

Consequences
Following Benefits are associated with this practice:

- Additional business expertise and higher chance of correct implementation. New connection to business needs for team, understand how the system is used by customer.
- Better understanding of LSADP on customer side. Higher appreciation through involvement.
- In some situations has lead to support for automation testing and accepting the smaller deliveries from customers as they understand the need for it.

Following Liabilities are associated with this practice:

- Feedback may be not directed at actual problem. Wasting time and increasing irritation.
- Disapproving customer can damage the relationship and the lower morale of team.
- Misuse can lead to damages

See Also
This practice is used in combination with **CO-04**: *Kickoff*, **CO-05**: *Product Backlog Refinement* and **CO-07**: *Periodic Round-Table* as events where the customer might be involved in.

C.1.6. Periodic Round-Table

CO-Pattern Overview	
ID	CO-07
Name	PERIODIC ROUND-TABLE
Alias	-
Summary	Recurring meeting of key users and Product Owners. The PO discusses and elicits functional requirements and business needs from the customer. Informs and updates the customer on current status of the functionality implemented for the user.

Example
Technology LLC. has a LSADP as an internal program to implement in-house software solutions. Hence, all customers of the Large-Scale Agile Development Program are internal customers representing several different business units and business needs. For aligning the creation process of requirements regular meetings are used between PO and customer groups. These are called "Periodic Round-Table".

Context
Whenever dealing with customers in software development the eliciting of the requirements is one of the most important steps in requirements engineering. In agile development especially an involved customer results in more accurately representations of the business need of a customer. However, customers have limited time and are not used to being regularly involved in the development process.

Problem
Following concerns are addressed by this practice:

- **C-15** :*How to elicit and refine requirements of end users?*
- **C-43** : *How to enforce customer involvement?*

Forces
Following Forces have been identified:

- Customers usually have a limited background in software development.
- Customers often have only an concept of a business need, without clear idea how the functionality should deliver solutions.
- Business Needs and the formed ideas for implementation need discussion as they seldom are the same. Hence, there is a need for compromises and flexibility on the customer-site.

Solution
Periodic Round-Table:
Recurring meeting every Sprint or every other Sprint depending on the amount of requirements a customer has. Periodic Round-Table with the respective key users and customers from the affected business units. In a first step, the customer describes a business need of his. In combination with the PO responsible for the module, they than collaborate to define a clear vision of the functionality. If the requirement comes with a major change or includes a high amount of work effort. There is a one or two day workshop between the PO and the customers involved. Creating a common understanding of the business need and the possibilities available to implement the requirement. After these initial discussions, in the Periodic Round-Table the customer then presents the business needs and functionality to the PO, to make sure they have the same understanding of what is required. The PO then explains a possible solution to the customer. In the following or the same Periodic Round-Table, the alignment of the requirement and the possible implementation leads to a, from both parties, confirmed finalized requirement. Usually it takes one to two Periodic Round-Table to finalize a requirement. The size and amount of people represented in a Periodic Round-Table, usually two to ten, depends on the amount of customers involved.

Consequences
Following Benefits have been associated with this practice:

- Common understanding of scope of a requirement, elicited in close cooperation.
- Close Relationship between PO and customer/key user.
- Customer is more involved.

Following Liabilities have been associated with this practice:

- Feedback or questions might be reserved until next Periodic Round-Table.
- Issues should be discussed as soon as the become obvious, the Periodic Round-Table hinder immediate discussions.

See Also
The Periodic Round-Table is often used to elicit **M-04**: *Acceptance Criteria* from the customer.

C.1.7. Process Consultant Meeting

CO-Pattern Overview	
ID	CO-09
Name	PROCESS CONSULTANT MEETING
Alias	-
Summary	Weekly meeting of Process Consultant and architects aligning architecture and designing module architecture. Further, talking about impediments in teams.

Example
Technology LLC. works with Purpose Teams and Process Consultants to align architectural decisions of all teams in the Large-Scale Agile Development Program they use Process Consultant Meetings with the IT-Architect **Context**
Aligning architecture over several domain-specific teams is difficult, as different impediments could arise or different mentalities need to be dealt with. Teams can have individual solutions, which need communication, so other teams are aware of interfaces within the system. And understand possible issues that can arise.

Problem
Following concerns are addressed by this practice:

- **C-26** :*How to align and communicate architectural decisions?*
- **C-85** :*How to share domain knowledge across agile teams?*

Forces
Following Forces have been identified:

- Clear Architecture strategy and situational architecture decisions have impact on the system, however full understanding of all system parts and modules is necessary.
- Working in several geographically distributed locations in a purpose team setup, can create communication shortages when considering architecture.
- Legacy systems with technical debts and limited to none test coverage, increase the possibility of unexpected errors.

Solution
Process Consultant Meetings:
Parallel Meeting to Scrum of Scrums offering a platform for weekly alignment and discussion of possible solutions for Impediments. Platform for architecture discussions and decision-making. Weekly meeting to address architecture topics. Introduced and lead by

IT-architect to present architectural concepts. The presented ideas are then discussed by all Process Consultants with their respective module knowledge in mind. Afterwards, the impediments in current sprint are addressed to discuss on a technical level a solution opportunity or refactoring possibilities. Update style Guide after changes decided in Process Consultant Meeting.

Variants
An alternative variation can be a **CO-1** for architectural topics. Similar discussions possible on higher level than only the Large-Scale Agile Development Program.

Consequences
Following Benefits have been associated with this practice:

- Increased knowledge pool as all domain experts are present and able to discuss architecture.
- Architectural Alignment
- Whole Product Focus when deciding on architecture
- Knowledge share across agile teams and opportunity to address impediments with other teams solutions

Following Liabilities have been associated with this practice:

- Excluding development team. Hence, another knowledge transfer required by Process Consultant.
- Possibly nontransparent decision-making, excluding other program members

C.1.8. Go-Live Celebration

CO-Pattern Overview	
ID	CO-10
Name	GO-LIVE CELEBRATION
Alias	-
Summary	Celebrate Go-Lives with teams involved. Make video-call in geographically distributed teams and block out time to acknowledge achievement.

Example
Product Owner (PO-6) realized that due to the deliver-driven nature of the Large-Scale Agile Development Program of Technology LLC. the actual successes are difficult to be measured, because after each delivery comes the next one.

Context
When working in a delivery focused setup, you might ignore the fact something has be delivered and just keep developing. With large goals, the focus on the achievements is hard to keep up and the morale might be negatively influenced.

Problem
Following concerns are addressed by this practice:

- **C-71** : *How to measure the success of the large-scale agile development program?*

Forces
Following forces have been identified:

- With geographical distribution, team coherence and team events are difficult to schedule
- What defines enough progress or delivery to be celebrated. When too little deliveries are celebrated, the outcome could be the opposite of what is the goal

Solution
Go-Live celebrations:
Focused on actual Go-Lives, no beta testing go-live or similar, as measure for large enough requirement so the celebration is adequate. Bring together all team members at the same location, once the go-live is done, preferably done during the time frame all teams are at work. Acknowledge the success of delivering the requirement, optionally have a little pizza celebration or similar. . Additionally, contact platform team to make time for short mention of go-live in review or other LSADP wide meetings. Take extra time at the end of

the year Review and in Kickoff to celebrate success stories of the team.

Consequences
Following Benefits have been associated with this practice:

- Shift focus from delivery-driven and make achievements visual
- Improved team coherence both on-location and across locations

Following Liabilities have been associated with this practice:

- Difficult to correctly involve geographically distributed teams
- Teams at different locations might feel left out, when one location is overdoing the go-live celebration

C.1.9. Third Party Interface-Planning Meeting

CO-Pattern Overview	
ID	CO-11
Name	THIRD PARTY INTERFACE-PLANNING MEETING
Alias	
Summary	Plan ahead for all interfaces with external parties, create a plan for their developments to align own development accordingly when required to adapt interface.

Example
Technology LLC.calls data provided by a third party system. Within their system the use the then provided data to calculate and work with it. However, information on updates or changes from third parties has to be elicited. Hence, the PO responsible for hird party systems makes planning meetings with external companies.

Context
Two projects working with different methods, one uses agile methodologies and the other works with a rigorous planning method.

Problem
The following concerns are addressed by this practice:

- **C-20** *How to facilitate communication between agile teams and other teams using traditional practices?*

Forces
The following forces have been identified:

- Interfaces to third parties need constant tracking, as soon as one end-point changes, the other has to be informed and adapt accordingly.
- Different work-styles can be difficult. Especially when unplanned events happen, is the response different, yet has to be aligned.

Solution
Third Party Interface-Planning Meeting:
Organize your interfaces and dependencies to third parties. Plan ahead in the Epic plan and make sure they are also aware of changes you are making to your system and which changes that might require on their side.

1. Create Epic Plan with focus on dependencies

2. Inform third parties or other projects as early as possible of the necessary changes which are to be done.
3. Before the requirements are part of Sprint hold initial communication summarizing all the important changes.
4. Stay in constant contact during the implementation phase.

Consequences
The following Benefits have been associated with this practice:

- Communication is enabled . quicker adaption and anticipation of their interface changes.
- Implementation is less challenging when adapting partly to other method.

The following Liabilities have been associated with this practice:

- Loss of agility required -No short-term changes possible.
- Higher work and communication load during sprints with interface topics.

See Also
Planning of these meetings is enabled through **V-06** *Epic Plan Game Board*. Be aware of **A-04** *Don't Assume mutual Terminology*.

C.2. Documentation of Good Methodology Practices

The good methodology practices **M-12** *Shifting Responsibilities* and **M-14** *Docupedia for Architectural Documentation* were already presented in the findings in Section4.4.4.

C.2.1. Definition of Ready and Definition of Done**

M-Pattern Overview	
ID	M-05
Name	DEFINITION OF READY AND DEFINITION OF DONE
Alias	-
Summary	Definitions by each team, to align understanding of what work Is required to take backlog item into development and what defines a "done" requirement.

Example
Technology LLC. has each team of LSAD define DoR and DoD to define thresholds that need to be met for requirement to be developed and then accepted as Done.

Context
Decreasing the undefined work required for a development team to be able to develop a requirement and for a the team to define a developed requirement as ready to be delivered.

Problem
Following concerns are addressed by this practice:

- **C-10** :*How to create precise requirement specifications for the development team?*
- **C-69** :*How to establish requirements verification?*

Forces
Following forces have been identified:

- Developers need the correct information and to understand the actual requirement and business need.
- Finished functionality needs to adhere to standards to improve customer satisfaction.

Solution
Definition of Ready and Definition of Done:
These are maintained by the agile teams and created during the Kickoff. DoR: Ticket has estimation and clarified business value to development team. Ticket has images of the screen to be changed and mock-ups. Acceptance Criteria are specified by PO,...
Any work to be done before a ticket can go into development.
DoD: Implementation is done. Functionality has been tested and peer-reviewed. test case documentation and tests are done,...
Any work to be done before ticket is closed.

Consequences
The following Benefits have been associated with this practice:

- clear definition and understanding for test and development of a requirement at all different stakeholders.
- Less necessity for queries. Customer and PO know what to expect.

The following Liabilities have been associated with this practice:

- Have to be adhered by otherwise useless.

See Also
Often combine with **M-04** *Acceptance Criteria*.

C.2.2. Reserved Capacity

M-Pattern Overview	
ID	M-01
Name	RESERVED CAPACITY
Alias	-
Summary	Calculation for each stakeholder groups capacity are done with a reduced amount of the actual available time. Reserving time for unplanned work and eventual bug fixes.

Example
Technology LLC. has a project team working in a LSADP. The team is delivery-driven and as a high throughput and many different modules, they support.

Context
Whenever a customer reports Bugs, which are minor and not breaking the system. The PO has to deal with a lower customer satisfaction. Hence, in planning a sprint he has to decide whether to focus on improving quality or keeping up with the known feature delivery speed. Explaining to the customer why both at once are hard to maintain.

Problem
The following concerns are addressed by this practice:

- **C-41** :*How to deal with unplanned requirements and risks?*
- **C-79** :*How to balance amount and quality of delivered requirements?*

Forces
Following forces have been identified:

- Customers expect a high feature development speed. At the same time, they want high quality products.
- When having a high throughput of features, chances are higher that Bugs appear which are not system breaking, however, bugs reported by customers can lead to an expectation they will be fixed right away.

Solution
Reserved Capacity:
Only calculate with 80% of the available capacity for the sprint. The other 20 % can be applied at the discretion of the PO to address unresolved Bugs or to deal with Blockers. Offering a degree of flexibility to enforce Quality relevant development.

Consequences
The following Benefits have been associated with this practice:

- Important bug fixes can be resolved in a sprint. Focus on quality assurance increases.
- More flexible in case blockers appear.

No Liabilities have been associated with this practice.

C.2.3. Scope Change

M-Pattern Overview	
ID	M-02
Name	SCOPE CHANGE
Alias	-
Summary	When the Sprint Goal or overall capacity is influenced by an unplanned risk or the loss of some development capacity, a scope change allows to redefine the achievable scope and keeps the team motivated.

Example
Scrum Masters at Technology LLC. have identified that changing the Scope is a appropriate measure when dealing with critical bug fixes or other impediments/risks during a Sprint. Keeping the Team more motivated.

Context
Requirements changed by customer or developer realizes the complexity of a requirement was underestimated. Or loss of development capacity leads to lower than expected deliveries.

Problem
Following concerns are addressed by this practice:

- **C-73** :*How to deal with decreased predictability?*

Forces
Following forces have been identified:

- When working with Legacy Systems and historically grown platforms predictions can turn out to be wrong more frequently.
- Limited or missing documentation or the required expert is no longer with the team, forces an requirement to take longer.

Solution
Scope Change:
The Team realizes a decreased predictability on a requirement or overall whether Sprint Goal is realizable. As a result, the team reassess in a PBR and in case the requirements become too complex, a Scope Change allows a reorganization of the current and following sprint. Freeing up some time to decide how to approach the issue. Team then decides together with platform team how to address issue.

Consequences
The following Benefits have been associated with this practice:

- Allows for reassessment and dealing with loss in predictability.
- Team still reaches sprint goal, which in turn motivates the team.

The following Liabilities have been associated with this practice:

- Can be misused, when used as a golden hammer solution.
- Less commitment to trying to achieve original goal, if overused.

C.2.4. Bug Prioritization

M-Pattern Overview	
ID	M-03
Name	BUG PRIORITIZATION
Alias	-
Summary	While working with a Product Owner Team in each Team, the PO responsible for a module with a Bug, has the Authority to prioritize the Bug as he likes, possibily changing the Sprint capacity.

Example
Technology LLC. works in a large-scale agile program, during the sprint planning or during a sprint one of the customers or a tester indicates a major bug. Leading to a Blocker or escalation. PO are all authorized to decide ion priority without consulting with other PO of team.

Context
Discovering Bugs, which can be system breaking, and deciding on the priority whether it needs to be resolved immediately.

Problem
Following concerns are addressed by this practice:

- **C-41** :*How to deal with unplanned requirements and risks?*
- **C-80** :*How to manage overarching backlog item prioritization with multiple product owners?*

Forces
Following forces have been identified:

- Unpredictable whether risks like a Bug will appear during a sprint. Also not avoidable.
- Which PO decides in a team on what requirement to focus on, especially when one has to deal with a unplanned risk

Solution
Bug Prioritization:
Product Owner has the authority to prioritize Bugs on their own. Unless a customer has triggered an escalation, the PO can decide on his or her own whether to include the new requirement in the sprint or to push it to a later sprint. When triggered by a tester unveiling a bug, the same process applies. Additionally the team has previously agreed to include

only Blockers in the same sprint. To address these unplanned occurrences more adequately all PO's have left capacity (20%) open in the team for each Sprint. The use of these 20% is at discretion of the PO **Consequences**
The following Benefits have been associated with this practice:

- No unnecessary discussions and Critical bugs get resolved.

The following Liabilities have been associated with this practice:

- Other POs' customer may be unhappy as a functionality of his won't be resolved in planned Sprint

See Also
This practices is used in combination with **M-11** *Product Owner Team* and the **]CO-01** *Pre-Planning Coordination*.

C.2.5. Acceptance Criteria

M-Pattern Overview	
ID	M-04
Name	ACCEPTANCE CRITERIA
Alias	-
Summary	Acceptance Criteria are used to set criteria for accepting a requirement. They are included with the backlog item and express the main functionalities needed for the item.

Example
Technology LLC. works in a large-scale agile development program. The PO has to define precise requirements specifications so the development team can implement the respective feature. For that he uses Acceptance Criteria.

Context
When creating requirements for the development team, the PO needs to specify the business needs of a customer on a level so the development team can create a fitting implementation. Avoiding wrong implementations as a core goal.

Problem
The following concerns are addressed by this practice:

- **C-10** :*How to create precise requirement specifications for the development team?*
- **C-69** :*How to establish requirements verification?*

Forces
Following forces have been identified:

- Developers need the correct information, which can be difficult to explain, to understand the actual requirement and business need.
- How can a PO make sure that his requirement has been resolved correctly in a manageable manor

Solution
Acceptance Criteria:
Concept for the ticket system. Each Ticket consists of detailed description what needs to be implemented to fulfill certain criteria, called acceptance criteria. The Acceptance Criteria define in detail what has to be resolved within the ticket.
' Ticket x Acceptance Criteria

- A drop-down with multiple vendor options needs to be added above button.
- The button has to deliver a Risk Analysis for the vendor selected in drop-down.
- No action on button click, when no vendor in drop-down selected.'

When all criteria are fulfilled, the Ticket is resolved

Variants
The use of **M-05** *Definition of Ready and Definition of Done* is a possible variation, but they can also be used together. **Consequences**
The following Benefits have been associated with this practice:

- Clear definition and understanding for test and development of a requirement at all different stakeholders. - Less necessity for queries.
- Customer and PO know where to expect change.

The following Liabilities have been associated with this practice:

- More time consuming than normal ticket creation

See Also
This practice is used in combination with **V-03***JIRA Board* and can complement the **M-05** *Definition of Ready and Definition of Done*.

C.2.6. Functional Splitting

M-Pattern Overview	
ID	M-06
Name	FUNCTIONAL SPLITTING
Alias	-
Summary	To avoid any confusion in when to split a backlog item, PO-3 at SADP of Technology LLC. Splits backlog items in the ticket-system according to the required function. One function = one ticket.

Example
Technology LLC. has a large-scale agile development program. A customer triggers a change or request for an additional feature, which comprises in a large requirement with several functional changes and the PO has to make the requirement understandable for the development team.

Context
Migration of legacy systems, other major change requests on existing modules or new feature development.

Problem
Following concerns are addressed by this practice:

- **C-18** :*How to split large and complex requirements into smaller requirements?*

Forces
Following forces have been identified:

- Difficult to constitute when splitting of requirements is required. Usually a requirement represents the backend logic and the frontend trigger for the logic.
- For large and complex requirements it's difficult to find a fitting split point. Which can make a requirement hard to understand.

Solution
Functional Splitting:
Functional splitting of requirements. When creating a requirement with a customer, the PO will note an overall required requirement, which he then will split into all required function changes or new implementations as a single requirement, When combining all these requirements the large and complex requirement is the result. Focus on only having one functional change or development for each requirement. If requested by the team

even splitting between front and backend logic. Creating a package of frontend requirement and backend requirement, which combined will resolve the larger requirement.

Consequences
The following Benefits have been associated with this practice:

- Smaller easier to understand requirements.
- Complexity reduction through splitting.

The following Liabilities have been associated with this practice:

- Extra effort and deep knowledge from PO required to identify all required functional changes and splitting them accordingly

See Also
Used in combination with **CO-05** *Product Backlog Refinement* to elicit when a split might be required with DEV.

C.2.7. Process Consultant

M-Pattern Overview	
ID	M-07
Name	PROCESS CONSULTANT
Alias	-
Summary	Process Consultants combine responsibilities of the roles of a developer, architect and consultant. The role finds application in LSAD with domain-specific knowledge. It can be compared to the concept of an Lead Developer. The Process Consultant aids in architecture decision making, knowledge sharing with less experienced developers and Product Owners and more. The Process Consultant is the most experienced developer of a domain responsible for the support in development of requirements as well as development itself.

Example
Technology LLC identified during the adoption process of a LSAD framework, the need for a Process Consultant. Process Consultants (Lead Developer) are domain experts supporting the development process of an system consisting of migrated legacy systems with inexperienced developers working on it.

Context
Complex and historically grown system. Migrating several legacy systems and providing complex functions. Project growth leads to inexperienced developers working on the system.

Problem
Following concerns are addressed by this practice:

- **C-19** :*How to deal with internal silos?*
- **C-85** :*How to share domain knowledge across teams?*

Forces
Following forces have been identified:

- High pace feature-driven development with newly created teams.
- Internal Silos and other bottlenecks in respect to knowledge as the teams work on legacy systems with complex functions and dependencies.

Solution
Process Consultant:
Process Consultant is like a technical lead for a specific domain and product. Creating an access point to knowledge by assigning a experienced developer, who can help less experienced developers and Product Owners understand system specifications and support creation process of new features and feature change. The Process Consultant is a 50/50 role, half of his time is allocated in supporting other developers and the PO work with the process. The other half is allocated for implementations and supporting architectural conception, alignment and implementation.

Variants
The Program uses experienced developers for multiple topics(architecture and team support, coaching for specific architecture topics, like UI/UX and Testing **CO-1***Community of Practice*.

Consequences
The following Benefits have been associated with this practice:

- Offers solution for knowledge holes by forcefully creating internal silos.
- Clear Contact Point for developers and PO alike through technical responsible person for domain.
- Accountability for product and domain from technical aspect.
- Efficiency Improvement with inexperienced developers. No stoppage during development process .
- PC introduction reduces bottlenecks by freeing up the day of these bottleneck to play a supportive knowledge transfer role.

The following Liabilities have been associated with this practice:

- Limits the functional knowledge and process of gaining functional knowledge of developers.
- Internal silos.

See Also
The practices is used in combination with **CO-09** *Process Consultant Meeting*, the **M-08** *Purpose Teams* and for clarity of dependencies **V-01** *Dependency Matrix*.

C.2.8. Purpose Teams

M-Pattern Overview	
ID	M-08
Name	PURPOSE TEAMS
Alias	-
Summary	Technology LLC uses Purpose Teams in their SADP as a solution for the complexity of the system, Purpose teams are an evolution of feature tems, including POs into the team and being assigned domains depending on knowledge expertise of team members.

Example
Technology LLC. introduced Purpose Teams to deal with inexperienced developers working on a complex system, regularly migrating and working on legacy systems.

Context
Complex system with complex dependencies and migrated legacy system supporting several modules, which are critical and need regular improvement as well as maintenance.

Problem
Following concerns are addressed by this practice:

- **C-19** :*How to deal with internal silos?*
- **C-85** :*How to share domain knowledge across teams?*

Forces
Following forces have been identified:

- Feature-driven development with lots of external pressure on quick resolution of requirements as well as maintenance of existing systems.
- Multiple key users and to deal with that multiple POs are introduced, PO has to be more visual in team.

Solution
Purpose Teams:
Purpose Teams are used to address specific bottlenecks in Knowledge Transfer. The Teams are organized with domain-specific solution and knowledge capabilities in mind. Breaking with the idea of co-location and avoidance of internal silos. Purpose teams actively create internal silos with knowledge transfer capabilities as a temporary solution to a larger knowledge transfer issue. With this concept, teams consist of all personnel required to

create a effective development team as well as customer support. A team is organized as follows and introduces following roles:

1. Modules are clustered depending on available domain knowledge
2. According to clusters, personnel responsible assigned to team.
   - Hence, a Team consists of several POs (each PO in a team has a module responsibility)
   - The role of Process Consultant**M-07** is introduced to address Knowledge Transfer issues
3. Teams are self-sufficient in developing in a Scrum format, all scrum team respective rules apply
4. Dependencies between teams are reduced as much as possible

Variants
A variation would be having full Feature Teams.

Consequences
The following Benefits have been associated with this practice:

- Solves short-term knowledge transfer issue.
- Allows for feature development and a incremental creation of several domain experts.
- Widens bottleneck with exact limit on affected personnel.

The following Liabilities have been associated with this practice:

- Requirements are assigned depending on supported module .
- More overhead hence single PO structure not a possibility.

See Also
This practice is used in combination with the introduction of **M-07** *Process Consultant* and the **M-11** *Product Owner Team*. With the practice of **M-12***Shifting Responsibilities* the internal silos are tried to be reduced.

C.2.9. Story Points

M-Pattern Overview	
ID	M-09
Name	STORY POINTS
Alias	-
Summary	Story Points are used as measure to estimate User stories, where as one Story Point represents the amount of work combined to create a story point. Hence, User Story complex need more Story Points than less complex User Stories. Team decides what a story point represents in amount of work.

Example
Team D of Technology LLC. uses Story Points as a leftover from Planning Poker to have a common metric for estimating user stories.

Context
When working agile, the importance of estimations becomes more significant as they provide a platform for discussion as well as a rough vision on the required development time. When working in a large-scale agile setup, a common estimation metric becomes more relevant as it provides a comparable observational metric.

Problem
Following concerns are addressed by this practice:

- **C-60**:*How to create and estimate user stories?*

Forces
Following forces have been identified:

- Reliable estimations require in-depth knowledge about the platform somebody is working on/ the domain they are working in. Large-scale agile teams form different conceptions of a estimation and a effort taken to fulfill a requirement.
- Person-Days still are a metric used in many IT-Organizations so LSAD have to figure out a way to communicate the actual effort required, which often is not reasonably quantifiable with Person-Days

Solution
Story Points:

1. Introduce Story Points as a common metric to be used in all teams.

2. No clear-cut LSADP wide definition on what a Story Point entails, simply a quantifiable method to draw relations between different requirements, hence, only used within teams.
3. When in a PBR the PO presents the requirement and the team assigns a Story Point value from Fibonacci formula (1, 2, 3, 5, 8, 13 ...).
4. A Requirement might require more effort as another requirement, hence the story point value is higher.
    - Example: Requirement 1 was assigned a Story Point value of 1
    - After describing Requirement 2 the team assigns a Story Point value of 2 as roughly twice the effort is required for the implementation.
5. Revisiting changing Requirements in PBR to keep Story Point values accurate in case an adaption is necessary.
6. Visualize Story Point Burn down to create visual of Process in several Teams for Review

Variants
Story Points can also be estimated by Expert Judgment,

Consequences
The following Benefits have been associated with this practice:

- Allows for a quick estimation purely based on relational estimation.
- Visualization of work done and work required possible.
- Formation of common understanding of required working effort a Story Point represents within a team.

The following Liabilities have been associated with this practice:

- Working with different teams the concept of one story point may change(i.e. Agile Coach).
- Moving requirements between teams may be an issue when using different scales for story points.

See Also
Always applied for estimation during **CO-05** *Product Backlog Refinement* and also documented in **V-07** *Velocity Sheet*.

C.2.10. Subtask-Testing

M-Pattern Overview	
ID	M-10
Name	SUBTASK-TESTING
Alias	-
Summary	Using the Subtask functionality in JIRA for creating relationship between tests and requirements.

Example
Technology LLC. uses Jira for requirement tracking. As a built in solution, of the JIRA-Tool, subtasks allow a direct allocation of tickets/tasks to each other. Company x uses these subtasks to indicated required test cases for a requirement and to track tests directly with requirements.

Context
Tracking tests and requirements difficult, especially when test cases are not connected to requirement and errors occur.

Problem
Following concerns are addressed by this practice:

- **C-53**:*How to ensure traceability of tests and requirements?*

Forces
Following forces have been identified:

- Handoff required between development and testing.
- Making sure the tests have been done.

Solution
Subtask-Testing:
Include Subtasks in Requirements to define required test cases as well as trace existing extensions to automated test. Done by tester- to trace which tests are done and how they look-, developers- to describe which test cases make sense- and product owners – to indicate what has to be reproduced in bug fixes and what to test for. Additional creation of test case documentation, traceability and test case definition and execution for each sprint assigned to tester within team.

Consequences
The following Benefits have been associated with this practice:

- Common understanding which tests are required.
- Easy traceability through build-in tool features.

No Liabilities have been associated with this practice.

See Also
In combination with a Tracking Board enabling subtasks i.e. **V-03** *JIRA Board*.

C.2.11. Product Owner Team

M-Pattern Overview	
ID	M-11
Name	PRODUCT OWNER TEAM
Alias	-
Summary	With multiple PO in one Team at Technology LLC. , each purpose team consists of a PO Team, with an Head PO for each Team.

Example
Technology LLC. has multiple Product Owner (PO) in a large scaled agile development program, to accommodate multiple customers. The roles of Head PO, in regards of a single team and overarching all teams a Chief Product Owner have been introduced to avoid any blockage.

Context
Multiple internal or external customers supported on multiple modules in an overarching system. Working with multiple teams, so one PO can no longer accommodate all modules and customers.

Problem
Following concerns are addressed by this practice:

- **C-35** :*How to define clear and visible priorities?*
- **C-80** :*How to manage overarching backlog item prioritization with multiple product owners?*

Forces
Following forces have been identified:

- A Product Owners purpose is to focus on his assigned products/modules. When sharing the capacity of a single development team over multiple Product Owners, the prioritization becomes important, as higher prioritized requirements get resolved sooner.
- A behavior of POs pitching the priority levels higher than necessary, to guaranty delivery for their modules' requirements.

Solution
Product Owner Team:
The roles of Head Product Owner and Chief Product Owner CPO) introduced to focus on the importance of the overarching program success. The Head Product Owner has the final

decision power over which requirements are to be resolved during a sprint. In his role, the Head PO coordinates with all POs involved in a team and nourishes a healthy cooperation between them. Overarching all teams the CPO has responsibility for the whole product supported by all teams, who are part of the large-scale agile development program. The CPO deals with scope and release decisions **Consequences**
The following Benefits have been associated with this practice:

- Transparent decision and responsibility assignment.
- Coordination improvement through regulatory force within teams.
- Focus on balance, so needs and requirements of all customers get addressed.

The following Liabilities have been associated with this practice:

- Introduction of authority levels among Product Owners.

See Also
This practice is combined with **M-03** *Bug Prioritization* and the concept of **M-08** *Purpose Teams*, also **CO-01** *Pre-Planning Meetings* are used to address coordination.

Other Standards
LeSS Huge advocates for Product Owner Teams

C.2.12. Automation Lead

MPattern Overview	
ID	M-13
Name	AUTOMATION LEAD
Alias	Test Architect
Summary	A Test Architect also called Automation Lead, who focusees on supporting all teams in creating automation tests.

Example
Technology LLC. works with and large-scale agile development program. To establish automated testing, they introduced an Automation Lead role. The Automation lead purpose is to monitor the level of automated testing in all teams as well as support everyone having issues with automated testing. Together with the Architects, he helps creating a vision and architecture concepts in regards to automated testing.

Context
Whenever migrating legacy systems or creating new code, testing helps to make sure the code works as expected. Instead of effort-high manual tests, automated testing offers a constant monitoring of important test cases.

Problem
Following concerns are addressed by this practice:

- **C-36** :*How to establish automated testing?*
- **C-68** :*How to write understandable automated tests?*

Forces
Following forces have been identified:

- Knowledge on correct set up of automated tests is limited.
- Developers often are overwhelmed with delivery date pressure and testing or gathering knowledge on correct automated testing falls short.
- Legacy code makes it hard to understand to purpose of some code blocks, limiting the amount of sufficient tests.

Solution
Automation Lead:
The automation lead is responsible to foster the automated testing capabilities of the agile program. By supporting developers and creating architecture to support the automated

testing. Additionally his responsible in shaping a mindset of quality assurance in the agile program. Supporting all agile program employees when creating automated tests and leading COP on testing. Finally the automation lead offers a Process Consultant aspect to all testers in a team.

Consequences
The following Benefits have been associated with this practice:

- Single point of contact with impediment solving capabilities.
- Support for architects in vision creation and implementation.
- Transparent vision and commitment to automated testing.

The following Liabilities have been associated with this practice:

- Loss of developer capabilities.

See Also
The Automation Lead is responsible for pushing agenda on quality assurance like **M-10** *Subtask-Testing*.

C.2.13. Impact Analysis

M-Pattern Overview	
ID	M-15
Name	IMPACT ANALYSIS
Alias	-
Summary	Performing an impact analysis to see what changes will do to legacy code.

Example
Technology LLC. works with legacy systems and migrates older systems into a new system. To avoid additional issues and help developers understand legacy code better, they save some time for an impact analysis when required.

Context
When a developer wants to make changes to a legacy system, he might not understand to a sufficient degree, how these changes might influence the working code around it.

Problem
Following concerns are addressed by this practice:

- **C-40** :*How to apply agile practices for developing or maintaining legacy systems?*

Forces
Following forces have been identified:

- Experts on legacy system might not be available anymore.
- Knowledge about functionalities within these systems can be limited.

Solution
Impact Analysis:
When refactoring is not possible during a sprint: Impact Analysis to understand legacy code before working on a requirement changing legacy code. Check all special cases described in legacy code to see full impact. Work through legacy code and understand it first, before changing any function in the legacy code. Making sure to note where the function is used and making sure to see all uses of the function. Creating an overview where a change in legacy code might create an impact. During the change on legacy code focus on testing all functionalities identified during impact analysis and then include a detailed check in the technical sprint as a new requirement.

Consequences
The following Benefits have been associated with this practice:

- Reduces bugs in the future .
- Saves time and allows to then fix easier in the future.

The following Liabilities have been associated with this practice:

- Can damage other functions that might be overlooked in impact analysis-(especially when working on generic module used by multiple other modules).

See Also
Often applied when doing a **M-18** *Proof of Concept*.

C.2.14. Incremental On-Boarding

M-Pattern Overview	
ID	M-16
Name	INCREMENTAL ON-BOARDING
Alias	-
Summary	Handing over increasingly difficult tasks to developers until On-Boarding is done.

Example
Technology LL's PCs use incremental On-Boarding to allow new employees with less experience to first understand the system and slowly be involved in the actual implementation.

Context
Highly complex system with several different domains and special domain knowledge. New employees have difficulty to gain enough deep knowledge from simply jumping in and participating in implementation.

Problem
Following concerns are addressed by this practice:

- **C-81** :*How to understand all interfaces and dependencies of the system?*

Forces
Following forces have been identified:

- On-Boarding is only limited in time frame, as new employees are needed to help development.
- different stakeholder groups need to gather different type of knowledge from On-Boarding.
- Complex System difficult to understand without working on it.

Solution
Incremental On-Boarding:
New developers are eased into their development role, by being mentored and only after a period of high involvement with the PC being assigned development tasks.

1. Set-Up development environment via Docupedia how-to and learn style guide.
2. KT sessions with PC, if possible KT-Workshops.
3. Assignment to Testing capabilities, strong contact with PC to understand necessity of tests and what should be monitored.

4. When new developer feels confident enough and PC confirms, first Assignment to development tasks.

Consequences
The following Benefits have been associated with this practice:

- Reduces test work of other developers in implementation phase.
- Better understanding among new developers.
- Focus on importance of testing–>Culture and mindset share.

The following Liabilities have been associated with this practice:

- Slower usability of new employees.
- Time effort of PC limited for other developers as he is required to help new employees.

See Also
Implemented to avoid **A-03** *Don't assume autonomous On-Boarding*, the KT is associated with **CO-02** *Face to Face Knowledge Transfer*, usually supported by **M-07** *Process Consultant*.

C.2.15. Planning Poker light

M-Pattern Overview	
ID	M-17
Name	PLANNING POKER LIGHT
Alias	-
Summary	Reduced Planning Poker to estimating user stories and direct discussions.

Example
Technology LLC. uses Planning Poker light to agree on estimating the amount of story points required.

Context
Working on a requirement might need different amount of time by different developers, however, when working in a large-scale agile development program, estimating these requirements becomes important as it helps planning the amount of work associated with a sprint or iteration.

Problem
Following concerns are addressed by this practice:

- **C-60** :*How to create and estimate user stories ?*

Forces
Following forces have been identified:

- Different experiences and knowledge on a domain lead to different estimations for requirements which are part of an iteration/sprint.
- Without reliable estimations it becomes difficult to plan a sprint as the amount of time required for a requirement might be overestimated/underestimated and in turn might block the development of other requirements

Solution
Planning Poker light:
Planning poker light(using video-chat tool): Golden rule: Stick to allocated time of PBR and as soon as time has been used, push estimation to next PBR

1. PO presents to be estimated requirement in PBR.
2. Short discussion in case any questions from development team.
3. Everybody types estimated Story Points in chat.

- Consensus done.
- Differences, short explanation by domain expert, what are the functional issues one can face, repeat 3 until consensus.

4. Repeat 3 for as long as time is left in PBR and requirements need estimation

Variants
Planning Poker as this is only the light adaption.

Consequences
The following Benefits have been associated with this practice:

- Quicker and less complex than planning poker.
- Team has common understanding of requirement. Focus on experienced estimations.
- As several PBR are done in one sprint.
- More focus on saying the actual amount of story points than all the framework around it.

The following Liabilities have been associated with this practice:

- Difficult to estimate in complex domain for less experienced developers.

See Also
Always applied at **CO-05** *Product Backlog Refinement*.

C.2.16. Proof of Concept

M-Pattern Overview	
ID	M-18
Name	PROOF OF CONCEPT
Alias	-
Summary	To check whether a requirement is realizable as specified by PO, a PC performs a Proof of Concept.

Example
Technology LLC. uses a Process Consultant and PO combination to address technical and business expertise issues. In case of an issue a Proof of Concept will be created by the PC to check whether the implementation of a business need is possible.

Context
Complex system with legacy code and domain specific expertise knowledge.

Problem
Following concerns are addressed by this practice:

- **C-63** :*How to explain requirements to different stakeholders?*

Forces
Following forces have been identified:

- Large requirements affecting legacy system are difficult for a PO to estimate on his own regarding implementation possibility
- Domain knowledge and limitations are not known by PO.

Solution
Proof of Concept:
PO adds requirement to sprint. Team and Process Consultant have concerns about possibility to fulfill the requirement. Process Consultant(PC) and PO discuss requirement and PC creates Proof of Concept(PoC). Proof of Concept:

1. PC creates a PoC for requirement which can't be discussed during a PBR or Sprint planning as it's too complex.
2. Checks whether the explained requirement is possible to be implemented.
   - Definition of criteria for success.
   - Check system whether these can be achieved with code possible and available.

3. Informs PO about outcome of PoC.
    - If PoC fails the requirement is denied.
    - Otherwise, PoC proofs possible implementation and requirement is added to a sprint.

Consequences
The following Benefits have been associated with this practice:

- Creates an argument whether a requirement is doable.
- Helps customer to understand the complexity facing a requirement.

The following Liabilities have been associated with this practice:

- Extra effort required by domain-expert.

See Also
Usually goes hand in hand with a **M-15** *Impact Analysis*.

C.3. Documentation of Good Viewpoint Practices

The good viewpoint practices **V-01** *Dependency Matrix* and **V-06** *Epic Plan Game Board* were already presented in the findings in Section4.4.4.

C.3.1. Burn-Down Chart

V-Pattern Overview	
ID	V-02
Name	BURN-DOWN-CHART
Alias	-
Summary	A Burn-Down Chart visualizes the amount of work done during a Sprint and helps to analyze how much work is left and when work was done. Eventually identifying tendencies of work done during a sprint.
V-Type	Report

Example
Technology LLC. works with a large-scale agile program. When describing requirements and user stories the company uses story points to quantify the actual complexity and required effort to implement a requirement.

Context
Measuring the success of a agile team is hard, therefore different methodologies are used to quantify the success of a sprint. To gain a level of understanding between several agile teams, a common methodology is necessary to provide indication of the success of a large-scale agile development program .

Problem
Following concerns are addressed by this practice:

- **C-71** :*How to measure the success of the large-scale agile development program?*

Forces
Following forces have been identified:

- Complexity of delivered features is difficulty to quantify by simply using text descriptions.
- The morale of a team is important and nurtured by visualizing the success of a team. When looking, from the outside, on a team it is hard to estimate purely from the requirements delivered, how successful a sprint was.

Solution
Burn-Down Chart:
Use of a quantifiable unit as Story Points to indicate the effort required. By establishing a common understanding what one unit of story point entails, the overall output becomes

visual. Then delivering this image via a burn down-chart. As well as a visualized overview over the quantified requirements, in story points(see FigureC.1 and FigureC.2), and in turn how many where totally estimated and resolved.

Consequences

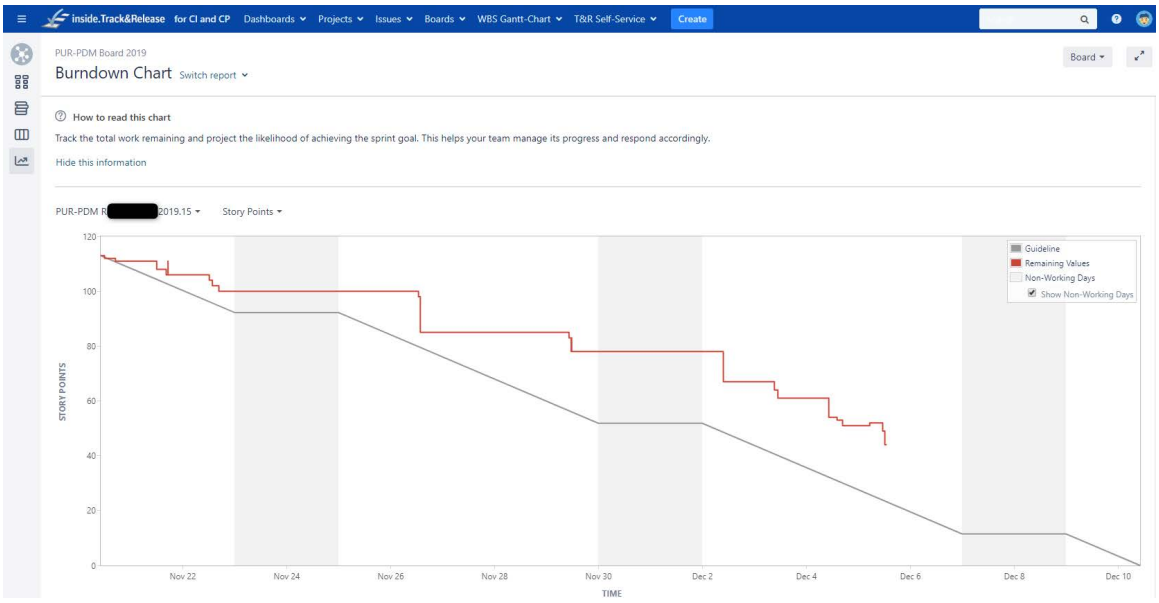


Figure C.1.: Exemplary Burn-Down Chart at Technology LLC.

The following Benefits have been associated with this practice:

- Measure for effort.
- Identifying and analyzing work done during sprint.
- Transparent visual for the process within a sprint or of older sprints.

The following Liabilities have been associated with this practice:

- Misuse of KPIs can hinder the productivity of a LSAD.

See Also

Used through te tool **V-03** *JIRA Board* and applying the measure of **M-09** *Story Points*.

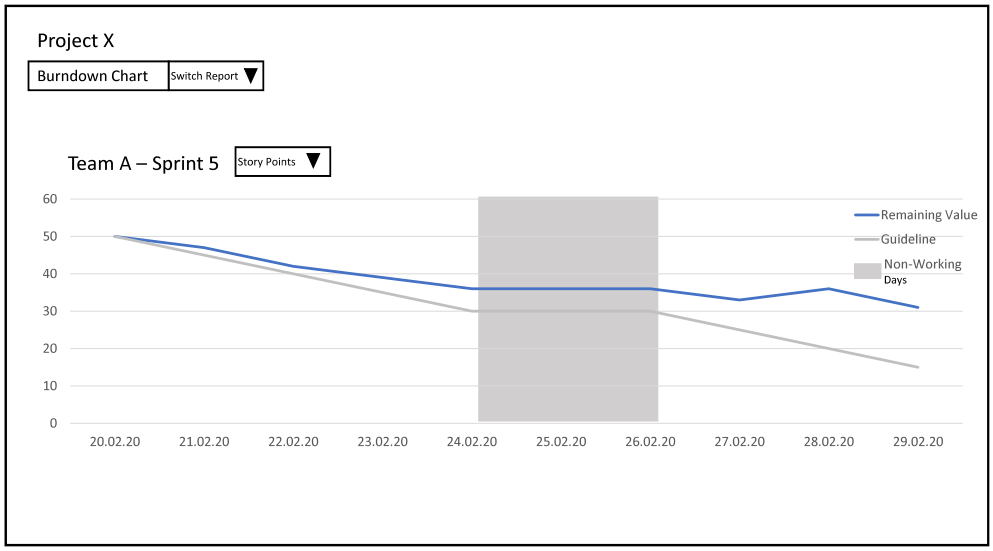


Figure C.2.: Burn-Down Chart Model

C.3.2. Jira Board

V-Pattern Overview	
ID	V-03
Name	JIRA BOARD
Alias	-
Summary	A ticket-based tool visualizing all requirements within a project. Allowing the tracking of statuses and the overall process of a requirement, from initiation to fulfillment.
V-Type	Board

Example
Technology LLC. works in a large-scale agile development program based on the LeSS framework. To keep track of all requirements in the Backlog, consisting of requirements and support tickets. Since the teams work geographically distributed, they use a JIRA Board.

Context
Managing several requirements of multiple teams. Visualizing requirements and support tickets in a ticket format providing additional information to each ticket as help to create solution.

Problem
Following concerns are addressed by this practice:

- **C-10** :*How to create precise requirement specifications for the development team?*
- **C-53** :*How to ensure traceability of tests and requirements?*

Forces
Following forces have been identified:

- Due to working in several locations, keeping a synchronized dashboard via an non-digital agile board is difficult.Creates a new level of responsibility, who is responsible to keep these non-digital boards all in the same state.
- Compare progress and see state of Sprint, whether it is challenged or on track.

Solution
JIRA Board:
Using a JIRA-Board (see FigureC.3 and FigureC.4) is like keeping a digital agile board, which is always in the same state for all users, independent of location. A digital Board like JIRA provides a ticket-based representation of all requirements. The tickets provide

some metadata on the requirement as well as any provided information via categories and detail description. On top, the tickets are movable over the board into different statuses, offering an instant overview for any team member. JIRA as a tool also provides visualizations for several views, such as burn down charts, sprint detail view etc **Variants**

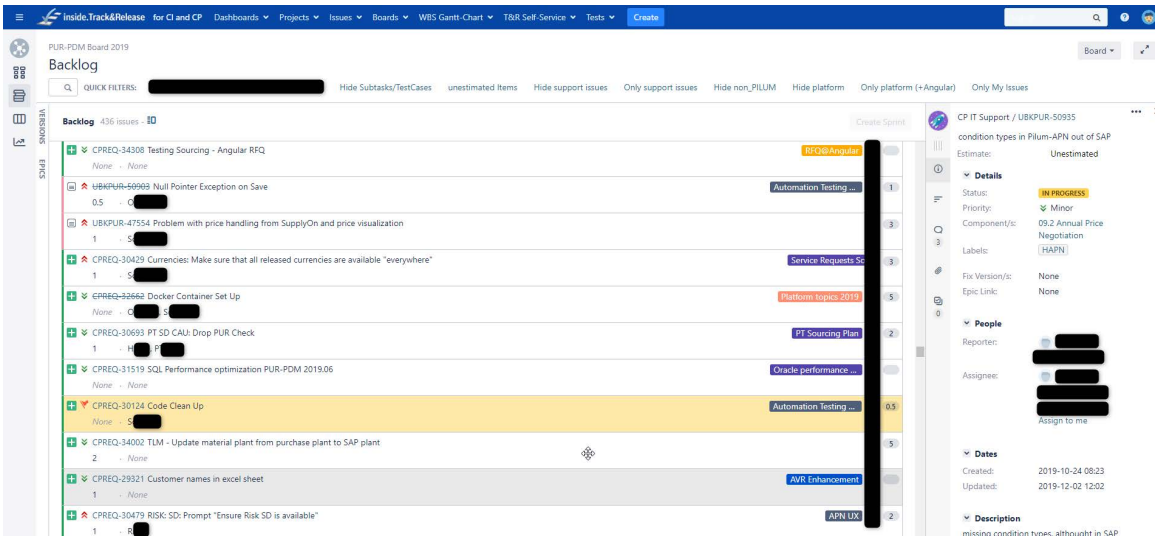


Figure C.3.: JIRA Board at Technology LLC.

Other similar ticketing tools or with co-located teams non-digital board.

Consequences
The following Benefits have been associated with this practice:

- Traceability of all requirements. Helps managing to-do's for all team members.
- Common comprehension for whole project. Additional analysis opportunities provided by tool.
- Clear documentation of all requirements. Transparent view on all sprints and capacities for all team members.

The following Liabilities have been associated with this practice:

- Different views can irritate, as they often look similar, but have different meaning.

See Also
IS used with the methods of **M-02** *Scope Change* in case, there are challenges and **M-10** *Subtask-Testing* to keep track of tests and requirements.

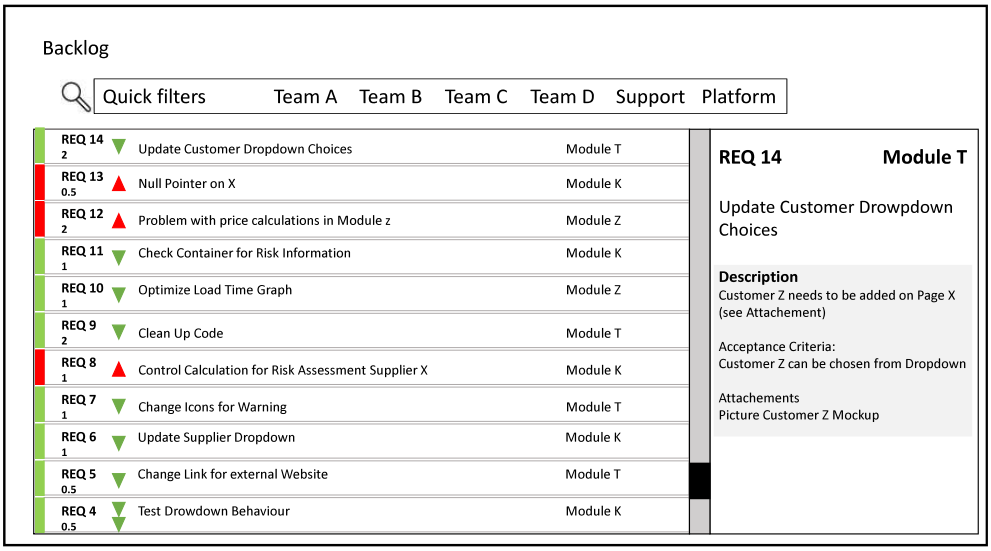


Figure C.4.: JIRA Board Model

C.3.3. Interface Architecture

V-Pattern Overview	
ID	V-04
Name	INTERFACE ARCHITECTURE
Alias	-
Summary	Documentation of all interfaces of a module.
V-Type	Model

Example
Technology LLC. uses docupedia for knowledge transfer. The project team has an gliffy uml-graph representing all modules supported and communication between these on a macro level.

Context
When working in a complex system on a specific module, cross-functionality can be hard to understand. Gathering knowledge on why and how modules interact cis similarly hard to communicate/memorize especially when working with legacy systems.

Problem
Following concerns are addressed by this practice:

- **C-26** :*How to align and communicate architectural decisions?*
- **C-81**:*How to understand all interfaces and dependencies of the system?*

Forces
Following forces have been identified:

- Legacy systems and complexity complicate cross-functionalities and the cooperation between several modules.
- Possibly making it hard to communicate the use of data models or even where some type of data set is used outside of one's knowledge

Solution
Interface Architecture:
Follow these Steps:

1. Creating a wiki entry for the architectural conception of project team
2. Setting project system in relation with overarching architecture landscape
3. Listing all supported modules by project team (compare Figure4.3)

4. Describing core functionality for all modules – focusing to not go into too much detail
5. Visualizing dependencies in a macro-architecture view
6. Visualizing interface, with getter/setter description, connection and synchronization schedule

compare to Figure APN

Consequences
The following Benefits have been associated with this practice:

- Visual of architectural landscape and exemplary interface communication.
- Description for application architecture.

The following Liabilities have been associated with this practice:

- Issues if not up-to-date.
- Too detailed may lead to more confusion.

See Also
M-14 *Docupedia for Architecture Documentation* for method on how to generate Views.

C.3.4. Power BI

V-Pattern Overview	
ID	V-05
Name	POWER BI DASHBOARD
Alias	-
Summary	Power BI offers as a tool a more in detail overview of the current Sprint progress. It additionally supports coordination tasks between multiple POs.
V-Type	Board

Example
Technology LLC. uses Microsoft Power BI Dashboard to offer a granular view on requirements, budget and capacity for all sprints. The view provided is available for all Product Owners and allows for a transparent communication and coordination among multiple POs.

Context
A large-scale agile program works with multiple POs in a team, who have to coordinate the requirements added to a sprint.

Problem
Following concerns are addressed by this practice:

- **C-35** :*How to define clear and visible priorities?*
- **C-80**How to manage overarching backlog item prioritization with multiple product owners?

Forces
Following forces have been identified:

- Multiple POs with responsibility for modules and different customers.
- Tracking of development outside of status and in relation to other requirements related. .

Solution
Power BI:
Power BI dashboard offers a specification for each requirement written by the PO responsible. The dashboard visualizes all specifications(see FigureC.5), sprints. modules, requirements and budgets in a singular view. Allowing for a perspective for all POs with a summarized view. Additionally calculations of developer days and supporting a team

perspective is available in the Power BI Dashboard.

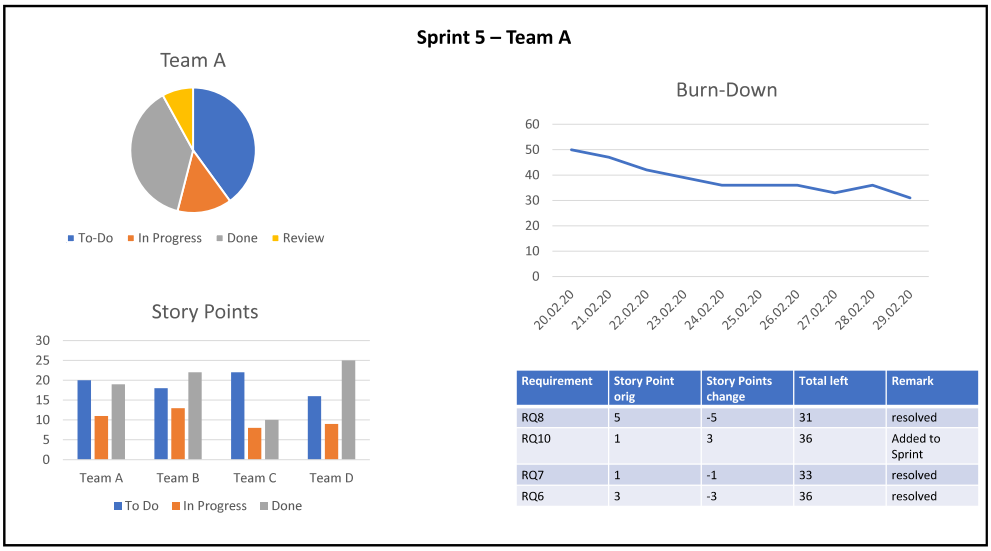


Figure C.5.: Power BI Model

Consequences
The following Benefits have been associated with this practice:

- Transparency for all POs and team. -Track future requirements and possibilities.
- Allows for planning and more transparent commitments to customers and POs. -Budget and Sprints are used here.
- Overview for POs.

The following Liabilities have been associated with this practice:

- Once daily updated data, could lead to calculations with old data.

See Also
USed in combination with **V-03** *JIRA Board* and **V-06** *Epic Plan Game Board* to keep track of all sprints and Budgets.

C.3.5. Velocity Sheet

V-Pattern Overview	
ID	V-01
Name	VELOCITY SHEET
Alias	
Summary	Technology LLC uses Velocity Sheets, to manage the capacity and potential throughput of projects. The velocity sheet is managed by the Product Owner and consists all team members, who in turn manage their own entries. According to the capacity, it calculates the velocity of a Team and allows for simpler estimations and planning for the Product Owners.
V-Type	Document

Example
Technology LLC has a project team working in a large-scale agile development program. To measure the available capacity of the developers, testers and other team members, they use a velocity excel-sheet.

Context
Capacity planning in regards to a sprint as well as costs for the customer and budget for the agile program.

Problem
Following concerns are addressed by this practice:

- **C-54** :*How to make a cost and schedule estimation?*

Forces
Following forces have been identified:

- Estimation on requirements have to be as accurate as possible.
- Different levels of developer skills, hence different time taken for same implementation by different developers.
- Measuring availability of team members in geographically distributed teams

Solution
Velocity Sheet:
The program approach is to calculate everything into capacity, so every requirement estimation delivers an amount of capacity required to resolve this requirement. Simultaneously each cost estimation is done via capacity estimation, so once a PO knows the capacity

required for a requirement and the available capacity for a sprint they could create a plan for the sprint and reply with an cost estimation for the customer. Hence, by managing an open Excel(see FiguresC.6 and C.7) where all team members are maintaining their capacity during a sprint. The capacity on that excel delivers everything required to both formulate cost and schedule estimations. Each development role has an defined amount of maximum capacity per day, so they only have to maintain the days they are available(i.e Developer 0.8, Process Consultant 0.5,)

1. Each team members responsibility to keep own capacity up-to-date
2. Representation of Availability of all team members
3. Representation of achieved Story Points in last Sprints
4. Moving Average of Story Points
5. Estimation of achievable Story Points for future Sprints via Velocity Sheet on basis of 1. – 3.

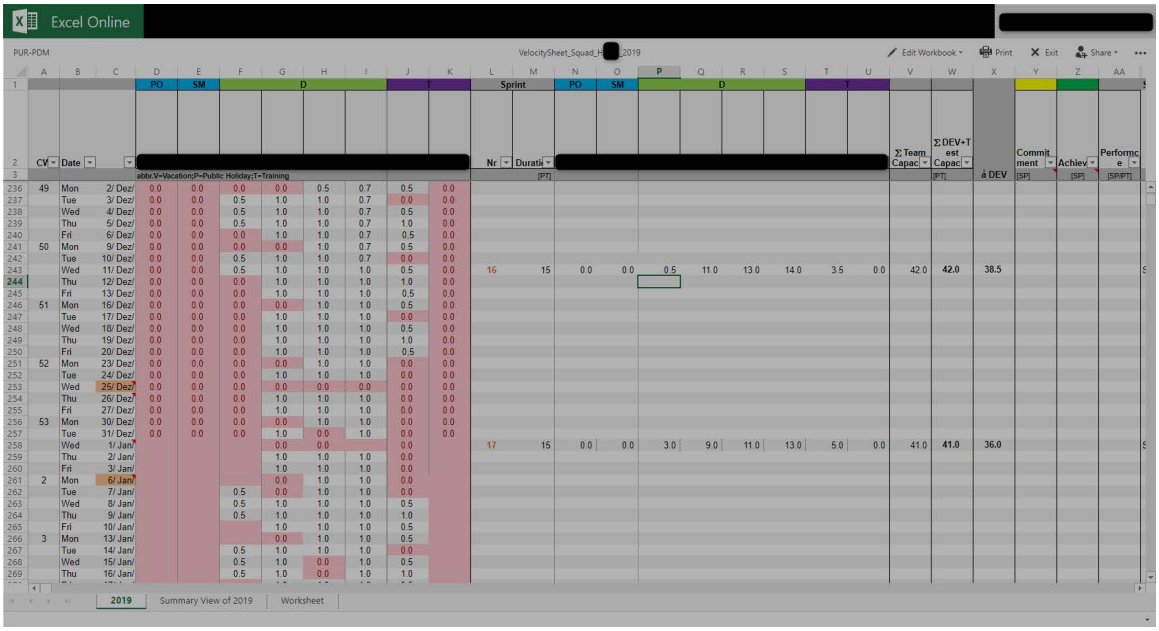


Figure C.6.: Velocity Sheet for exemplary Team at Technology LLC.

Consequences
The following Benefits have been associated with this practice:

- One key figure is only required and all team members help maintain it as well as allow communication to different stakeholders outside the project.

CW	Date		PO	SM	DEV			Test	Sprint	Team Capa	DEV Capa	Commit ment
6	Mon	3/Feb	0.5	0.5	0.5	1	1	1	2	40.5	23.0	18.5
	Tue	4/Feb	0.5	0.5	0.5	1	1	1				
	Wed	5/Feb	0.5	0.5	0.5	1	1	1				
	Thu	6/Feb	0	0.5	0.5	1	1	1				
	Fri	7/Feb	0.5	0.5	0.5	0	1	0				
7	Mon	10/Feb	1	0.5	0.5	0	1	1				
	Tue	11/Feb	1	0.5	0.5	1	1	1				
	Wed	12/Feb	0.5	0.5	0.5	1	1	1				
	Thu	13/Feb	0	0.5	0.5	1	1	1				
	Fri	14/Feb	0	0.5	0.5	1	1	0				

Figure C.7.: Velocity Sheet Model

The following Liabilities have been associated with this practice:

- If not properly maintained problems in exactly this communication.

See Also
Used to organize and support **CO-05** *Product Backlog Refinement*.

C.4. Documentation of Principle Candidates

The principle candidate **P-01** *Geographically Distributed Meeting Hours* was already presented in the findings in Section4.4.4.

C.4.1. Avoid extra meetings

Principle Overview	
ID	P-02
Name	AVOID EXTRA MEETINGS
Alias	-
Summary	Technology LLC defines meetings that are necessary for the development process. Outside of these meetings, direct communication is preferred and the guideline is to avoid adding extra meetings outside of the agreed meetings within a project.
Type	Communication & Coordination
Binding Nature	Recommended

Example
Technology LLC. works with a large-scale agile development program and has per team an amount of preset meetings to fulfill a successful development process. In addition, the team members of the program agreed to don't have meetings outside of the set of meetings preplanned in the morning.

Context
Cross-shore working program and teams who have to communicate and coordinate details about requirements, while working.

Problem
Following concerns are addressed by this principle-candidate:

- **C-87** :*How to clarify details outside of meetings in cross-shore agile teams?*

Forces
Following forces have been identified:

- Queries about requirements or impediments which would take too long for a daily or similar meeting, need communication, however, with different time zones it becomes more difficult to apply.
- Geographical Distribution makes it difficult to communicate directly, with different working hours.

Solution
No additional meetings outside of already confirmed meetings

- Being available for the team outside of structured meetings and reserve some time after meetings for clarification

- Directly contact people when having any questions
- If absence write e-mail or comment in task(ticket in jira)
- Documentation of decisions also in tickets, making them visible to all team members
- Visualization of decisions made during development

Known Uses
Technology LLC.

Consequences
The following Benefits have been associated with this principle-candidate:

- Direct communication saves time and is more efficient

The following Liabilities have been associated with this principle-candidate:

- Work on multiple tickets waiting on busy colleagues response

See Also
In combination with **P-01** *Geographically Distributed Meeting Hours*.

C.4.2. Semi Co-Location

Principle Overview	
ID	P-03
Name	SEMI CO-LOCATION
Alias	-
Summary	Technology LLC. has geographically distributed teams, however, they focus on having at least two team members with the same role at the same location. Allowing for better communication and additional coordination when facing impediments.
Type	Communication & Coordination
Binding Nature	Recommended

Example
Technology LLC works with a large-scale agile development program and has per team at least two developers at the same location to allow for semi co-located workspaces.

Context
Cross-shore working program and teams who have to communicate and coordinate. Teams are working in different locations and each location consists of a role in the large-scale agile program.

Problem
Following concerns are addressed by this principle-candidate:

- **C-19** :*How to deal with internal silos?*

Forces
Following forces have been identified:

- Depending on the supported modules a team setup may vary in size and distribution according to the available budget.
- Geographical Distribution in LSADP

Solution
Semi Co-Located:
For each agile team, at least two persons of the same role are working at the same location. When not possible, have at least two persons in the same role, but in different teams located at same location.

Known Uses
Technology LLC.

Consequences
The following Benefits have been associated with this principle-candidate:

- Direct communication for same roles, contact in person to clarify issues

The following Liabilities have been associated with this principle-candidate:

- Difficult in setting up
- Creating a location coherence instead of a team coherence

C.5. Documentation of Anti-Pattern and Bad Practices

The Anti-Pattern **A-1***Don't use Frameworks as Recipe* was already presented in Section4.4.1. The bad practices **A-04** *Don't Assume mutual Terminology* and **A-07** *Don't Have New Years Resolution Dilemma* were already presented in the findings in Section4.4.12.

C.5.1. Don't overshoot Coordination Meetings**

Anti-Pattern Overview	
ID	A-06
Name	DON'T OVERSHOOT COORDINATION MEETINGS
Alias	-
Summary	Development Team accepts not fully prepared requirements to PBRs or Developers don't prepare themselves for meetings. Leading to prolonged and inefficient coordination meetings. Possibly needing more meetings than required.

Example
Technology LLC. has delivery pressure and starts skipping recurring meetings, pushing estimation etc back into a sprint planning meeting.

Context
Unexpected Delivery pressure or loss of labor power leading to slower delivery times.

Problem
Following concerns are addressed by this bad practice:

- **C-36** :*How to establish automated testing?*
- **C-63** :*How to explain requirements to stakeholders?*

Forces
Following forces have been identified:

- Due to time constraints, some tickets not prioritized before Sprint Planning.
- Cycle once behind; the team always appears to be a step behind each planned meeting.

General Form
Prioritization and Estimation of Requirements not completed until Sprint Planning. Hence, PO are part of Sprint Planning and discussion entails, resulting in a impromptu PBR instead of Sprint Planning and increasing the pressure on development team.

Consequences
The following Benefits have been associated with this bad practice:

- Estimation and prioritizing gets done.

The following Liabilities have been associated with this bad practice:

- Over commitment through pressure on development team
- Decreased Quality of delivered products

Revised Solution
Use of **CO-01** *Pre-Planning Coordination* and **M-05** *Defintion of Ready and Definition of Done*.

See Also
Also avoid **A-07** *Don't have New Years Resolution Dilemma*

C.5.2. Don't have Blurred Boundaries Requirements Engineering

Anti-Pattern Overview	
ID	A-01
Name	DON'T HAVE BLURRED BOUNDARIES REQUIREMENTS ENGINEERING
Alias	-
Summary	Product Owners tend to take over requirements engineering responsibilities of their customers, if the customer has limited or no experience on how to adequately describe business needs or even explain the required functionality efficiently.

Example
Technology LLC has multiple customers for their large-scale agile program. Some customers do not have the required skill set to accurately describe their business need, so the PO steps in and defines the business need for the customer.

Context
Customer without required skill set to properly define business needs.

Problem
Following concerns are addressed by this bad practice:

- **C-15** :*How to elicit and refine requirements of end users?*

Forces
Following forces have been identified:

- Requirements need to reflect the business need of customers, however, discussions can limit the plan of an agile program.
- In large companies there are individuals who get put on a position assuming they have a certain skill set, without ensuring this assumption is accurate. A certain degree of IT understanding is expected, but there is no clear role definition on customer site fort he contact persons role

General Form
Customer can't describe his requirements and the business need they try to represent. Additionally the customer has issues understanding technical limitations or has a grasp on the current status of the system. To accelerate the process, the PO starts taking over the construction on his own. Assuming the business need and requirements. Effectively blurring the line between actual requirement a customer needs and pays for and the generated

requirement of the PO. The program in turn does not know whether they are building the right application or if the customer actually would need another application. Since the customer was not involved in the requirements generation, the business need is potentially not covered.

Consequences
The following Benefits have been associated with this bad practice:

- Requirements created and the development can begin.

The following Liabilities have been associated with this bad practice:

- No security that implementation represents the actual business need of customer.
- Wrong communication on customer side, as they might expect something different than communicated with agile program

Revised Solution
Being patient in requirements creation and use **CO-07** *Periodic Round-Table* for building up skill set and explaining customer your needs.

C.5.3. Don't force Team Coherence

Anti-Pattern Overview	
ID	A-02
Name	DON'T FORCE TEAM COHERENCE
Alias	-
Summary	Scrum Masters are focused on creating a working team, which is able to deliver and has a preferably good team cohesion. However, some team members might not get along, yet, work well together. The Scrum Master has to notice this and not force a team coherence too far.

Example
Technology LLC has a project team with two members, who on a business level work well together, but they don't get along on a human level. The Scrum Master of the team sees this situation and tries to force a better team coherence .

Context
SM and the team notices two or more team members do not get along and often have arguments. Resulting in them not discussing work relevant information.

Problem
Following concerns are addressed by this bad practice:

- **C-44** :*How to deal with communication gaps with stakeholders?*

Forces
Following forces have been identified:

- Personalities can collide.
- Team cooperation and team coherence is important to achieve maximum efficiency

General Form
Scrum Master or any other role with focus on team coherence, tries to resolve arguments between team members. As a goal of reaching a good team chemistry to focus shifts too much to the personal side.

Consequences
The following Benefits have been associated with this bad practice:

- Involved parties discuss their issues and reach understanding.

The following Liabilities have been associated with this bad practice:

- Involved parties feel attacked or micromanaged by SM or other role who tries to resolve.
- Risk of broadening the gap between the team members.

Revised Solution
As long as the outcome is not affected and the team chemistry does not decrease, the SM or other team members can ignore the conflict.

C.5.4. Don't assume autonomous On-Boarding

Anti-Pattern Overview	
ID	A-03
Name	Don't assume autonomous On-Boarding
Alias	-
Summary	Large-Scale Agile Development Program uses a Wiki to support their On-Boarding process for several different stakeholder goups. However, the information provided isn't tailored to all stakeholder groups effected, rather only for a single stakeholder group. Nonetheless, autonomous On-Boarding is expected for all stakeholder groups.

Example
Technology LLC. has a large-scale agile program. Due to the increasing amount of work and managed modules, the program adds new Product Owners and other team members. To increase the time it takes for new team members to work in the project, the new employees handle the On-Boarding process on their own.

Context
When dealing with new employees in a large-scale agile program, that has been adapted to the team's needs. Getting employees up to speed.

Problem
Following concerns are addressed by this bad practice:

- **C-10** :*How to create precise requirement specifications for the development team?*
- **C-82** :*How to support an On-Boarding approach for different stakeholders?*

Forces
Following forces have been identified:

- At scale, the detailed documentation of the product and the generated and the On-Boarding of new employees becomes more important as the product itself has outgrown the current supporting team.
- Time for detail documentation as well as On-Boarding efforts is scarce.
- Complexity of platform and modules is high and hard to understand, because of several mitigated legacy systems, from the provided documentation. Provided Documentation cannot describe to a detail degree necessary for On-Boarding work.

General Form
After initial On-Boarding, administrative and meeting the colleagues. New employees need to work through months, maybe years of development on their own. Always with the focus on getting involved as soon as possible in the daily work of the teams. While providing a documentation, in form of a wiki, specifying the team setup, the agreements made by each team (DoR and DoD) and some architecture design. These information are limited to structure and organization of the program. Yet, not providing a clear role and stakeholder specific documentation. Omitting detail information for several processes. Leaving the new employee to figure out details like requirements engineering via trial and error.

Consequences
The following Benefits have been associated with this bad practice:

- already involved team member spends less time on the On-Boarding of a new employee. When described in detail the wiki can be sufficient to understand everything necessary for future work.

The following Liabilities have been associated with this bad practice:

- first couple of meetings, the actual involvement of new employees is low. The employee might not understand in detail how to do his job, as an example when requirement engineering the required information is not given and the meeting takes longer, wasting time of all involved employees.

Revised Solution
Addressed by **CO-1** *Community of Practice*, **M-07** *Process Consultant* and **M-12** *Shifting Responsibilities*

C.5.5. Don't forward Requirements

Anti-Pattern Overview	
ID	A-05
Name	DON'T FORWARD REQUIREMENTS
Alias	-
Summary	Product Owner trusts a key stakeholder in specifying requirements on his own and forwards the requirement without further editing and specification.

Example
Technology LLC. has a large-scale agile development program; all customers are internal business units. The Product Owner forwards requirements created by customers without editing them or having any discussion.

Context
Whenever dealing with customers in software development the eliciting of the requirements is one of the most important steps in requirements engineering, as it offers insight on the business need as well as a platform to discuss possible implementation possibilities.

Problem
Following concerns are addressed by this bad practice:

- **C-15** :*How to elicit and refining requirements of end users?*

Forces
Following forces have been identified:

- When dealing with multiple key users, a PO has a lot of work and when one of these key users is trusted and experienced, a inexperienced PO might assume that his specifications will be understandable.
- overwhelmed PO accepts the requirements send to him

General Form
Requirements are not refined with the customer, instead the customer's initial requirement definition is „forwarded" to the development team. PO is inexperienced and trusts a customer's expertise to describe a sufficient requirement. As a result the PO does not discuss and compromise with the customer on a doable and sufficient implementation, but instead simply "forwards" the requirement.

Consequences
The following Benefits have been associated with this bad practice:

- Requirement is elicited.

The following Liabilities have been associated with this bad practice:

- A insufficiently described requirement can lead to inefficient implementation and in the worst case create chaos as possible cross-functionalities might be affected.
- There are no real advantages as the PO simply omits one of his major tasks, so the time he saves and the customers wrong sense of completing a task and expecting a business need being resolved, are only short-lived. Usually the customer is dissatisfied with the delivered feature and the development process slows down as avoidable problems can appear

Revised Solution
Use of **CO-07** *Periodic Round-Table*

C.5.6. Don't limit KT to KT Workshops

Anti-Pattern Overview	
ID	A-08
Name	DON'T LIMIT KT TO KT WORKSHOPS
Alias	-
Summary	LSAD started by focusing Knowledge Transfer to be explicitly done during Workshops to avoid time waste. However, in a geographically distributed program, workshops were all members get together are difficult to organize.

Example
Technology LLC. supports and implements on a platform, which migrated several legacy systems. Developers working on the platform require in-depth domain specific knowledge.

Context
To support Knowledge Transfer, specific dates for KT workshops are set during the end of year business. Domain Experts present knowledge about domain.

Problem
Following concerns are addressed by this bad practice:

- **C-19** :*How to deal with internal silos?*
- **C-85** :*How to share domain knowledge across teams?*

Forces
Following forces have been identified:

- Complex Platform with vast amount of legacy code.
- Different skill-levels with developers and Process Consultants – technical lead developer for module or domain – are necessary to allow normal flow

General Form
Knowledge Transfer Workshops as a Golden Hammer solution for knowledge transfer. Process Experts provide a workshop on the domain they are an expert in, without having a real grasp on the knowledge level of participants. Participation is optional for all team members and actual participants may have different levels of knowledge. One time Process for vast knowledge transfer instead of incremental KT.

Consequences
The following Benefits have been associated with this bad practice:

- Specific use for On-Boarding purposes providing a good introduction to the platform.

The following Liabilities have been associated with this bad practice:

- Participants may need a baseline introduction and therefore feel overwhelmed.
- Presenter/Expert does not know whether there is any positive effect from his presentation.

Revised Solution
Addressed by **CO-1** *Community of Practice*, **M-07** *Process Consultant* and **M-12** *Shifting Responsibilities*

C.5.7. Don't misuse Estimation Creation

Anti-Pattern Overview	
ID	A-09
Name	DON'T MISUSE ESTIMATION CREATION
Alias	-
Summary	LSAD uses a methodology for estimating backlog items, however, they don't correctly implement and act on the methodology. Instead creating a prolonged process with limited usable outcome.

Example
Technology LLC. previously used a methodology like "Planning Poker", with an in length discussion which requirements need how much time. Through adaption, this methodology was deemed to lengthy and a reduction to estimating with story points via vote in a video-call chat was decided on.

Context
When estimating the amount of time required resolving a requirement, a project uses parts of methodologies to both speed up the estimation process, yet keep to a certain degree reliable estimations.

Problem
Following concerns are addressed by this bad practice:

- **C-60** :*How to create and estimating user stories?*
- **C-81** :*How to understand all interfaces of the architecture?*

Forces
Following forces have been identified:

- Adaption of a methodology, hence, there has been already a decision made to change a methodology to profit from reduction of complexity.
- The reduction leads to increasingly inaccurate estimations as well as a loss of value/importance to the estimation process.

General Form
When estimating a requirement, the estimation often is reliant on single developer. This single developer, from the other participants' perspective, is apparently making an estimation out of thin air. Additionally the downscaled methodology does not really offer a

platform for discussion during estimation. By associating a Story Point with a comparison-based methodology and then relying on a single developer's opinion, the group-based estimation aspect gets lost. Especially prolific in cross-team expertise communication as the team includes an expert from another domain with specific knowledge to help estimating cross-module influencing code changes. When a project has been adapting and adding different methodologies and roles to fit their way of work. The project has successfully been creating their individually fitting framework. This success indicates to the project team that all adaptions to a chosen framework work. Even though adaption is necessary, it can also start to erode the core principles associated with a framework.

Consequences
The following Benefits have been associated with this bad practice:

- Increased speed in Product Backlog Refinements and meetings in general by using a down-scaled process

The following Liabilities have been associated with this bad practice:

- Less accurate estimation leading to less throughput of the project.
- Chance of loss of customer trust or overworking employees as deliveries cannot be upheld.

Revised Solution
USe of **M-17** *Planning Poker light*.

C.5.8. Don't limit external colleagues access

Anti-Pattern Overview	
ID	A-10
Name	DON'T LIMIT EXTERNAL COLLEAGUES ACCESS
Alias	-
Summary	External and remotely working colleagues are limited in their capability to work, when not situated at a location.

Example
Technology LLC. has remotely working/ external colleagues involved in their large-scale agile development program. When working from outside the companies' network, security measures have implications on the quality of connection and possibility of connection.

Context
Company network limits the accessibility from remotely working/external colleagues .

Problem
Following concerns are addressed by this bad practice:

- **C-86** :*How to involve remotely working/external colleagues?*

Forces
Following forces have been identified:

- Company policy as well as laws prohibit certain forms of connection or data handling
- Company policy to include external organizations at certain scale of project.

General Form
When working with Skype for Business company policy as well as country specific law prohibits recording these calls. In addition, the connection from remote sources triggers technical impediments, lower audio quality and prohibits screen sharing by subjects from outside the companies' network. As well as sharing the screen to those sources.

Consequences
The following Benefits have been associated with this bad practice:

- Secure Company network

The following Liabilities have been associated with this bad practice:

- Delay in audio; no screen-sharing

Revised Solution
Workarounds exist, to mitigate the audio issue. Calling in by phone lowers the delay of communication for example. All in all communication flow must be supported, hence, having external work from in-house locations with company laptop as often as possible as main solution

C.5.9. Don't capsulate teams too much

Anti-Pattern Overview	
ID	A-11
Name	DON'T CAPSULATE TEAMS TOO MUCH
Alias	-
Summary	High demand for domain specific knowledge creates a need for highly specialized teams. However, these limit the KT between teams.

Example
Technology LLC. has a large-scale agile development program supporting many different modules and migrating multiple legacy systems. To keep up with delivery timelines the program switched to purpose teams with domain and module specific expertise.

Context
Domain and module specific teams supported by domain experts. Limiting the interaction between teams to required communication and coordination on interface level.

Problem
Following concerns are addressed by this bad practice:

- **C-19** :*How to deal with internal silo?*
- **C-85** :*How to share domain knowledge across agile teams?*

Forces
Following forces have been identified:

- Different domain projects and complex, hard understandable modules for each purpose teams.
- Contact to different program hard as the teams are both highly involved in own project

General Form
To keep up with delivery timelines the program switched to purpose teams with domain and module specific expertise. Purpose teams focused on domains and assigned modules. When a team member of one purpose team is assigned a module of another purpose team, it is hard to find ways of communications. By creating encapsulated teams with domain specific knowledge, the knowledge about this domain is also encapsulated in the team.

Consequences
No Benefits have been associated with this bad practice.
The following Liabilities have been associated with this bad practice:

- Prolonged communication efforts required.
- Knowledge Transfer for domain specific issue is limited within respective team

Revised Solution
Addressed by **CO-1** *Community of Practice*, **M-07** *Process Consultant* and **M-12** *Shifting Responsibilities*.

C.6. Documentation of Patterns for Implementation Process

The pattern **CO-1** *Community of Practice* and the pattern **P-1** *Celebrate Every Success* were already presented in the findings in Section4.5.1.

C.6.1. Supervision**

As documented by Uludağ and Matthes[57]

CO-Pattern Overview	
ID	CO-2
Name	SUPERVISION
Alias	-
Summary	A SUPERVISION offers agile teams a platform to discuss their current problems in a small and closed circle of participants and jointly find and evaluate solutions to these problems.

Example
A scrum master at ConglomerateCo is assigned to an agile team that has an over-cautious product owner that delays the start time of the first sprint. The scrum master is overwhelmed with this situation and looks for suitable solutions to deal with this problem. At ConglomerateCo, the scrum master does not have suitable platforms to discuss his problem with other scrum master and to ask for their personal experience on similar situations.

Context
Agile teams face a variety of problems in their daily work that go beyond actual implementation challenges that are not addressed in the retrospectives for time or confidentiality reasons. Furthermore, retrospectives do not provide a suitable platform to discuss domain-specific challenges with the same agile roles.

Problem
The following concerns are addressed by this pattern:

- **C-67** :*How to encourage development teams to talk about tasks and impediments?*

Forces
Following forces have been identified:

- Some employees do not like to talk openly about their problems in front of their colleagues.
- Some people do not want to raise problems with their colleagues when the people concerned are present to avoid bigger escalations.
- No suitable platforms are existing for discussing domain-specific problems with colleagues having equal roles.

Solution
Supervision:
Set up a SUPERVISION meeting with four to eight participants for at maximum three hours. A typical agenda of a SUPERVISION is structured as follows:

1. Casting: Every participant thinks of one to two problems he wants to discuss. Every problem is shortly introduced by each participant. Afterwards, the participants vote on which of the problems are going to be discussed in the current SUPERVISION. The two most frequently chosen problems are discussed in the later part of SUPERVISION.
2. Telling: The person who introduced the problem, called the storyteller, has to explain his problem in more detail. The other participants are not allowed to talk or to ask questions as long as the storyteller depicts his problem.
3. Asking: At this stage, participants can ask comprehension questions to better understand the problem.
4. Hypothesis: During this stage, The participants state some hypothesis on the problem. Here, the storyteller should be physically away from the other participants, e.g., by leaving the room or staying behind a flip chart, so that an intervention of the brainstorming participants is not possible. At this stage, the creativity process should not be disturbed by the storyteller.
5. Feedback: The storyteller evaluates the hypotheses.
6. Solution: The participants present solutions for addressing the stated problem.
7. Feedback: The storyteller evaluates the proposed solutions and explains which of them are feasible and which are not.

Variants
A SUPERVISION can be done within an agile team or on a cross-team level with people from the same domain. A domain-specific SUPERVISION can focus on typical problems of agile coaches, product owners, and architects.

Consequences
The following Benefits have been associated with this pattern:

- It provides a secure environment to talk about sensitive issues.
- Based on the experiences of the collective, well-structured solutions are proposed for the problems discussed.
- Participants can reflect on the problems and solutions addressed for their own work.
- Solutions to the problems are gathered by different people, resulting in a wider range of possible solutions with each different benefits and drawbacks.

The following Liabilities have been associated with this pattern:

- Problems that are irrelevant to the participants can be neglected.

- Participants might not feel valued if their problem is not discussed.
- In the case of communicating the discussed problems with other employees outside of this circle, it can lead to a breach of trust.

Known Uses
The following uses of this pattern are known:

- ConglomerateCo
- RetailCo
- SoftwareConsultCo

C.6.2. Communicate Architecture**

Adapted from Uludağ et al[58]

CO-Pattern Overview	
ID	CO-3
Name	COMMUNICATE ARCHITECTURE
Alias	-
Summary	Architects of other organizations prefer direct inter- and intra-team communication, through the introduction of the Communicate Architecture, there is a platform and measure for the architect to communicate the architecture directly to the teams.

Example
Technology LLC.s' platform team notices a gap between the architectural topics they are pursuing and the actual realization within the teams. Additionally, they noticed that the vision for the architecture and some general coding guidelines are not seen the same in all teams.

Context
With It-Architects and EA- Architects not being part of the agile teams of a LSADP it becomes more difficult to correctly align architecture and communicate any decisions made for architecture. The Architects need to find a method or platform for continued communication with single teams but also with all teams.

Problem
Following concerns are addressed with this pattern:

- **C-26**:*How to align and communicate architectural decisions?*
- **C-81**:*How to understand all interfaces and dependencies of the system?*

Forces
Following forces have been identified:

- Geographical distributed agile teams with small window of parallel working hours, which is already reserved for LSADP meetings.
- Specialized domain teams with multiple cultural backgrounds, therefore it is difficult to find a measure for communication with each team.

Solution
Communicate Architecture:
There are three main concepts for directly communicating architecture.

1. Find opportunities in the LSADP existing meetings to address architectural topics to all teams for intra-team communication.
2. Find opportunities to be involved in team specific meetings to address inter-team communication.
3. Be available for any queries about architecture and be proactive.

In addition the use of Process Consultants allows for additional alignment and communication by efficiently using **CO-09** *Process Consultant Meeting* to address measures of communication.

Consequences
Following Benefits have been associated with this pattern:

- Increased architectural alignment.
- Common Understanding of system and architectural vision.
- Closer and more effective communication channels for architects.

Following Liabilities have been associated with this pattern:

- Avoid meetings, where too many people are not interested or the goal of the meeting is different, as it would decrease the effect.

See Also
As already mentioned can be used by addressing architectural topics in **CO-08** *Newsflash* and Sprint Reviews, additionally using **CO-04** *Kickoff* for intra-team communication. For inter-team communication the **M-07** *Process Consultant* and the **CO-09** *Process Consultant Meeting* are useful.

C.6.3. DDD: Event Storming Workshops**

Adapted from Uludağ et al.[56]

C-Pattern Overview	
ID	CO-4
Name	DDD: EVENT STORMING WORKSHOPS
Alias	-
Summary	Event Storming Workshops are used to evolve the domain model, part of tactical DDD ubiquitous language to describe domain. Team and EA-Architect update domain model by focusing on all events of the domain an bringing them in relation, to find methods of updating these relationships or add new functionality elegantly.

Example
The EA of InsuranceCorp saw the necessity to introduce both data modeling on domain level and event storming to address solution and architectural alignment capabilities of InsuranceCorp. Event Storming Workshops are held with all team members of a domain, hence all team members are part of the solution generation.

Context Complex business domains with legacy systems are difficult to understand, but even more so when trying to update or migrate these systems into a new platform. Figuring out the business flow might help understand the systems in their entirety.

Problem
Following concerns are addressed by this pattern:

- **C-81** :*How to understand all interfaces and dependencies of the system?*
- **C-84** :*How to involve all team members in the solution generation?*

Forces
Following forces have been identified:

- Business logic of legacy systems without the expertise is difficult to understand.
- Finding individual architectural solutions for domain-teams and communicating them on the right platform.
- Involving the domain experts as to not create useless architecture for a domain

Solution
DDD: Event Storming Workshop:
The architect moderates the event storming workshop, most effective before the start and

after a epic finishes. Bringing together all domain experts and developers to work on the domain model. As the name indicates, this pattern is focused around events: 'Then the group adds the commands, or triggers, that cause the events, and considers all sources of commands, including users, external systems, and even time.'[34] The group focuses on the events associated with the domain and writes each event on a post-it, which will then be distributed according to their dependencies over the room[56]. Build a comprehensive business flow model with the post-its and then hold discussion about where issues are which events need to be addressed and where to involve new events to collaboratively design the new domain model and business flow[56]. After the event storming workshop, make sure to document the room and the findings regarding domain model and business flow and include it on a wiki-page[56].

Consequences
Following Benefits have been associated with this pattern:

- Common understanding of domain across whole team.
- more insight into the business flow and the Architect and his vision are clearer.

Following Liabilities have been associated with this pattern:

- Cost intensive when having to bring together geographically distributed teams.
- Good solution for remote workshops required.

C.6.4. Quality Gates**

Adopted from Uludağ and Matthes[60] and Pathania[41]

M-Pattern Overview	
ID	M-1
Name	QUALITY GATES
Alias	-
Summary	Addressing quality concerns by introducing Quality Gates in Jenkins using SonarQube, measures for code quality, defined by the Architects to be passed in the continuous delivery pipeline.

Example
Technology LLC.s' LSADP notices their focus on being delivery-driven, as several colleagues point out, the quality of code and the coverage of test cases needs improving.

Context
With several teams working on migrating legacy systems and developing new functionality for complex and critical systems, the quality of delivered functionality becomes more important. If issues are not recognized there might be a negative impact on the customers' satisfaction or even the system not behaving as it should.

Problem
Following concerns are addressed by this pattern:

- **C-36** :*How to establish automated testing?*
- **C-39**:*How to create a culture of continuous improvement?*

Forces
Following forces have been identified:

1. With focus on delivery, the goal of creating high quality systems, is pushed back.
2. The benefit of having automated checks of quality of code in the continuous delivery pipeline appears to be out-weight by the effort and necessity
3. With several epics per team per year, the amount of work to be done became the main focus.

Solution
Quality Gates:
Introducing quality gates to continuous delivery pipeline. In Jenkins, tools like SonarQube support quality gates, these can be integrated at several points of the pipeline. Each gate can be managed with several conditions, checking for complexity of a file/class, condition

coverage and coverage overall and many more[41]. For each condition a value can be assigned, which would trigger a Warning or Error. Additionally, for each condition a person group or single person, can be assigned, who will get notified when a condition throws an error. On notification, the code triggering th error will be stopped and needs a manual pull by one authorized person. With this technique, the assigned person can check the code committed and via manual pull, send it on in the pipeline.

Consequences
Following Benefits have been associated with this pattern:

- Code standard is increased as there is no code being delivered not passing a quality gate.
- Confidence in delivery is increased.
- Non-invasive measure, increasing focus on improving code quality.

Following Liabilities have been associated with this pattern:

- Extra responsibility for person being notified by quality gate, make sure as not to introduce unnecessary hierarchies .
- Make sure condition and value are not too high or low as it can influence the acceptance of the quality gate.

Bibliography

[1] Christopher Alexander. "The origins of pattern theory: The future of the theory, and the generation of a living world". In: *IEEE software* 16.5 (1999), pp. 71–82.

[2] Christopher Alexander. *The timeless way of building*. Vol. 1. New York: Oxford University Press, 1979.

[3] The LeSS Company B.V. *Graphics from the less.works website*. 2020. URL: `https://less.works/resources/graphics/site-graphics.html`.

[4] The LeSS Company B.V. *LeSS - More with LeSS*. 2020. URL: `https://less.works/`.

[5] S Balaji and M Sundararajan Murugaiyan. "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC". In: *International Journal of Information Technology and Business Management* 2.1 (2012), pp. 26–30.

[6] Julian M Bass. "Scrum master activities: process tailoring in large enterprise projects". In: *2014 IEEE 9th International Conference on Global Software Engineering*. IEEE. 2014, pp. 6–15.

[7] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. *The agile manifesto*. 2001.

[8] Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, and Jeff Sutherland. "SCRUM: An extension pattern language for hyperproductive software development". In: *Pattern languages of program design* 4 (1999), pp. 637–651.

[9] Ilya Bibik et al. "How to Kill the Scrum Monster". In: *Springer Books* (2018).

[10] Sabine Buckl, Florian Matthes, Alexander W Schneider, and Christian M Schweda. "Pattern-based design research–an iterative research method balancing rigor and relevance". In: *International Conference on Design Science Research in Information Systems*. Springer. 2013, pp. 73–87.

[11] Frank Buschmann, Regine Meunier, Peter Sommerlad, Michael Stal, and Hans Rohnert. *Pattern-oriented software architecture, Volume 1, A system of patterns*. 1996.

[12] Lan Cao and Balasubramaniam Ramesh. "Agile requirements engineering practices: An empirical study". In: *IEEE software* 25.1 (2008), pp. 60–67.

[13] James O Coplien and Gertrud Bjørnvig. *Lean architecture: for agile software development*. John Wiley & Sons, 2011.

[14] James O Coplien and Neil Harrison. *Organizational patterns of agile software development*. Pearson Prentice Hall Upper Saddle River, 2005.

[15] James O. Coplien and A Word On Alexander. *Software Patterns*. 1996.

[16] D.S. Cruzes and T. Dybøa. "Recommended Steps for Thematic Synthesis in Software Engineering". In: *2011 International Symposium on Empirical Software Engineering and Measurement*. 2011, pp. 275–284.

[17] Philipp Diebold, Jan-Peter Ostberg, Stefan Wagner, and Ulrich Zendler. "What do practitioners vary in using scrum?" In: *International Conference on Agile Software Development*. Springer. 2015, pp. 40–51.

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

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

[20] Torgeir Dingsøyr and Casper Lassenius. "Emerging themes in agile software development: Introduction to the special section on continuous value delivery". In: *Information and Software Technology* 77 (2016), pp. 56–60.

[21] Torgeir Dingsøyr and Nils Brede Moe. "Research challenges in large-scale agile software development". In: *ACM SIGSOFT Software Engineering Notes* 38.5 (2013), pp. 38–39.

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

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

[24] Alexander M Ernst. "A pattern-based approach to enterprise architecture management". PhD thesis. Technische Universität München, 2010.

[25] Christoph Fuchs and Thomas Hess. "Becoming agile in the digital transformation: the process of a large-scale agile transformation". In: (2018).

[26] Satoshi Hino. *Inside the mind of Toyota: Management principles for enduring growth*. CRC Press, 2005.

[27] VersionOne Inc. *13th annual state of agile development survey*. 2019. URL: `https://www.stateofagile.com/#ufh-c-473508-state-of-agile-report`.

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

[29] Maarit Laanti. "Characteristics and principles of scaled agile". In: *International Conference on Agile Software Development*. Springer. 2014, pp. 9–20.

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

[31] Craig Larman and Bas Vodde. *Large-scale scrum: More with LeSS*. Addison-Wesley Professional, 2017.

[32] Craig Larman and Bas Vodde. "Scaling lean & agile development". In: *Organization* 230.11 (2009).

[33] Jeffrey Liker. *The toyota way*. Esensi, 2004.

[34] Steven A. Lowe. *An introduction to event storming: The easy way to achieve domain-driven design*. 2020. URL: `https://techbeacon.com/devops/introduction-event-storming-easy-way-achieve-domain-driven-design`.

[35] Neil Maiden and Sara Jones. "Agile Requirements Can We Have Our Cake and Eat It Too?" In: *IEEE software* 27.3 (2010), pp. 87–88.

[36] Christoph Mathis. *SAFe–Das Scaled Agile Framework: Lean und Agile in großen Unternehmen skalieren. Mit einem Geleitwort von Dean Leffingwell. SAFe 4.5 inside*. dpunkt. verlag, 2018.

[37] Peter Measey. *Agile Foundations : Principles, Practices and Frameworks.* BCS, The Chartered Institute for IT, 2015.

[38] 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*. IEEE. 2013, pp. 119–123.

[39] Maria Paasivaara, Benjamin Behm, Casper Lassenius, and Minna Hallikainen. "Large-scale agile transformation at Ericsson: a case study". In: *Empirical Software Engineering* 23.5 (2018), pp. 2550–2596.

[40] Maria Paasivaara and Casper Lassenius. "Communities of practice in a large distributed agile software development organization–Case Ericsson". In: *Information and Software Technology* 56.12 (2014), pp. 1556–1577.

[41] Nikhil Pathania. *Learning continuous integration with Jenkins: a beginner's guide to implementing continuous integration and continuous delivery using Jenkins 2*. Packt Publishing Ltd, 2017.

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

[43] Ken Power. "A model for understanding when scaling agile is appropriate in large organizations". In: *International Conference on Agile Software Development*. Springer. 2014, pp. 83–92.

[44] Knut Rolland, Torgeir Dingsoyr, Brian Fitzgerald, and Klaas-Jan Stol. "Problematizing agile in the large: alternative assumptions for large-scale agile development". In: *39th International Conference on Information Systems*. Association for Information Systems (AIS). 2016.

[45] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical software engineering* 14.2 (2009), p. 131.

[46] Roman Sauter, Werner Sauter, Roland Wolfig, et al. *Agile Werte-und Kompetenzentwicklung*. Springer, 2018.

[47] Alexander W Schneider and Florian Matthes. "Evolving the eam pattern language". In: *Proceedings of the 20th European Conference on Pattern Languages of Programs*. 2015, pp. 1–11.

[48] Ken Schwaber and Jeff Sutherland. *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game*. 2017. URL: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf.

[49] Scrum.org. *What is Scrum*. URL: https://scrumorg-website-prod.s3.amazonaws.com/drupal/2016-06/ScrumFramework_17x11.pdf.

[50] ScrumPLoP. *ScrumPLoP - Published Patterns*. 2020. URL: https://sites.google.com/a/scrumplop.org/published-patterns/home.

[51] Gernot Starke and Peter Hruschka. *arc42-Template*. 2019. URL: https://arc42.org/overview/.

[52] Gernot Starke and Peter Hruschka. "Eine Strukturvorlage zur effektiven Dokumentation von Software-und IT Architekturen". In: *Wirtschaftinformatik Proceedings 2007* (2007), p. 61.

[53] Jeff Sutherland and Ken Schwaber. "The scrum papers". In: *Nuts, Bolts and Origins of an Agile Process* (2007).

[54] Inc The Standish Group International. *CHAOS Report 2015-Final*. 2015. URL: https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf.

[55] Ömer Uludağ, Nina-Mareike Harders, and Florian Matthes. "Documenting recurring concerns and patterns in large-scale agile development". In: *Proceedings of the 24th European Conference on Pattern Languages of Programs*. ACM. 2019, p. 27.

[56] Ömer Uludağ, Matheus Hauder, Martin Kleehaus, Christina Schimpfle, and Florian Matthes. "Supporting large-scale agile development with domain-driven design". In: *International Conference on Agile Software Development*. Springer. 2018, pp. 232–247.

[57] 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: *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE. 2018, pp. 191–197.

[58] Ömer Uludağ, Martin Kleehaus, Soner Erçelik, and Florian Matthes. "Using Social Network Analysis to Investigate the Collaboration Between Architects and Agile Teams: A Case Study of a Large-Scale Agile Development Program in a German Consumer Electronics Company". In: *International Conference on Agile Software Development*. Springer. 2019, pp. 137–153.

[59] Ömer Uludağ and Florian Matthes. "Identifying and Documenting Recurring Concerns and Best Practices of Agile Coaches and Scrum Masters in Large-Scale Agile Development". In: (2020).

[60] Ömer Uludağ and Florian Matthes. "Identifying and Documenting Recurring Concerns and Best Practices of Enterprise Architects and Solution Architects in Large-Scale Agile Development". In: (2020).

[61] Ömer Uludağ, Sinan Özgün, and Florian Matthes. *Recommender System for Scaling Agile Frameworks*. 2019. URL: `https://scaling-agile-hub.sebis.in.tum.de/#/patterns`.

[62] Vaughn Vernon. *Implementing domain-driven design*. Addison-Wesley, 2013.

[63] Xiaofeng Wang, Kieran Conboy, and Oisin Cawley. ""Leagile" software development: An experience report analysis of the application of lean approaches in agile software development". In: *Journal of Systems and Software* 85.6 (2012), pp. 1287–1299.

[64] Etienne Wenger, Richard Arnold McDermott, and William Snyder. *Cultivating communities of practice: A guide to managing knowledge*. Harvard Business Press, 2002.

[65] Laurie Williams and Alistair Cockburn. "Agile software development: it's about feedback and change". In: *IEEE computer* 36.6 (2003), pp. 39–43.

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