Identifying and Documenting Recurring Concerns and Best Practices of Agile Coaches and Scrum Masters in Large-Scale Agile Development

ÖMER ULUDAĞ, Technische Universität München
FLORIAN MATTHES, Technische Universität München

Ever since the release of the agile manifesto in 2001, agile methods have received widespread interest in industry and academia. Agile methods have transformed and brought unique changes to software development practices by strongly emphasizing team collaboration, change tolerance, and active customer involvement. Their proven benefits have also inspired organizations to apply them in large-scale settings. However, the adoption of agile methods at scale entails unique challenges such as coordinating and aligning multiple large-scale agile activities, dealing with internal silos, and establishing an agile culture & mindset throughout the organization. In particular, agile coaches and scrum masters are confronted with unprecedented concerns in large-scale agile development. Notwithstanding their importance for large-scale agile endeavors, extant literature still lacks an overview of their typical concerns and a collection of patterns to address them. Against this backdrop, we provide an overview of typical concerns and present five best practices of agile coaches and scrum masters in large-scale agile development.

Categories and Subject Descriptors: A.0 [**General**] Conference proceedings; K.6.3 [**Software Management**] Software Development

Additional Key Words and Phrases: agile coaches, concerns, large-scale agile development, patterns, scrum masters

ACM Reference Format:

Uludağ, Ö. and Matthes, F. 2019. Identifying and Documenting Recurring Concerns and Best Practices of Agile Coaches and Scrum Masters in Large-Scale Agile Development. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 26 (October 2019), 25 pages.

1 Introduction

Emerging in the 1990s, agile software development methods such as Extreme Programming [Beck 2000] and Scrum [Schwaber and Beedle 2001] have transformed and brought unprecedented advancements to software development practice by emphasizing change tolerance, team collaboration, and customer involvement [Kettunen 2007; Dingsøyr and Moe 2014]. With these methods, small, co-located, self-organizing teams work closely with the business customer on a single-project context, maximizing customer value and software product quality through rapid iterations and frequent feedback loops [Kettunen 2007]. Since agile methods have proved to be successful at the team level, large organizations are now aiming to scale agile methods to the enterprise level [Alqudah and Razali 2016]. Version One's 12th survey on the state of agile [VersionOne 2018] also reflects this industry trend towards adopting agile methods in-the-large. The survey shows that 52% of the 1492 respondents work in companies where the majority of teams are agile. However, the adoption of agile methods at larger scale entails

Author's address: Ö. Uludağ, Boltzmannstrasse 3, D-85748 Garching b. München; email: oemer.uludag@tum.de; F. Matthes, Boltzmannstrasse 3, D-85748 Garching b. München; email: matthes@tum.de

new challenges such as coordinating several large-scale agile activities, establishing an agile culture & mindset, and dealing with general resistances to changes [Dikert et al. 2016; Uludağ et al. 2018]. Especially agile coaches and scrum master are confronted with a number of unprecedented concerns in large-scale agile development [Uludağ et al. 2018]. Notwithstanding the significance of agile coaches and scrum masters for the success of large-scale agile endeavors, extant literature disregards an overview of their concerns and a collection of best practices to address them. Against this backdrop, we provide an overview of typical concerns of agile coaches and scrum masters and present five best practices.
The remainder of this paper is structured as follows. In Section 2, we portray the research design of our paper. In Section 3, we report on related works and describe related pattern languages. In Section 4, we elaborate the conceptual overview of the underlying pattern language to document recurring concerns and patterns. In Section 5, we give an overview of identified concerns and best practices. In Section 6, we discuss our main findings before concluding our paper with a summary of our results and remarks on future research in Section 7.

2 Research Approach

Our larger research initiative strives to document best practices that address recurring concerns of stakeholders in large-scale agile development. To balance the rigor and relevance of our research, we followed the pattern-based design research (PDR) method [Buckl et al. 2013]. The PDR method encourages researchers to theorize and learn from their intervention at industry partners while conducting rigorous and relevant design science research. The PDR method consists of four phases: *observe & conceptualize*, *pattern-based theory building & nexus instantiation*, *solution design & application*, and *evaluation & learning* (see Fig. 1).
During the *observe & conceptualize* phase, good practices for recurring concerns are observed and documented based on a typical pattern structure (see Section 4). These pattern candidates are then conceptualized by using grounding theories and evolve into genuine patterns by meeting the *rule of three*[1] [Coplien 1996], which are then integrated into the large-scale agile development pattern language. Pattern candidates, patterns, and the pattern language form an organized collection of reusable, proven solutions. In the *solution design & application* phase, stakeholders in large-scale agile development make the use of this knowledge base and select patterns based on their concerns. Selected patterns have to be configured and adjusted to the terminology of the company. After their configuration, adapted patterns can be used within the case organization. During the *evaluation & learning* phase, deviations between the actual and original pattern configuration are identified and documented. These deviations can be used to identify new best practices.

3 Related Work

Despite the industry trend towards adopting agile methods in-the-large [VersionOne 2018], sound academic research is lagging, especially regarding challenges and success factors [Dikert et al. 2016; Alsaqaf et al. 2019; Uludağ et al. 2018]. Some researchers witnessed this gap and started to publish academic papers, which are described in the following.
[Dikert et al. 2016] made a first attempt and reported 35 challenges and 29 success factors from 42 different organizations by conducting a systematic literature review of industrial large-scale agile transformations. The most salient challenge categories were *agile difficult to implement*, *integrating non-development functions*, *change resistance*, and *requirements engineering challenges*. The most important success factors were *management support*, *choosing and customizing the agile model*, *training and coaching*, and *mindset and alignment*. By means of a literature review and case study, [Kalenda et al. 2018] reported challenges and success factors of large companies adopting agile methods. They identified the following four challenges, namely *resistance to change*, *quality assurance issues*, *integrating with non-agile parts of the organization*, and *too fast roll-out*. Moreover, they

[1] The *rule of three* suggests that a documented pattern must refer to at least three known uses in practice to guarantee the re-usability of the given solution.

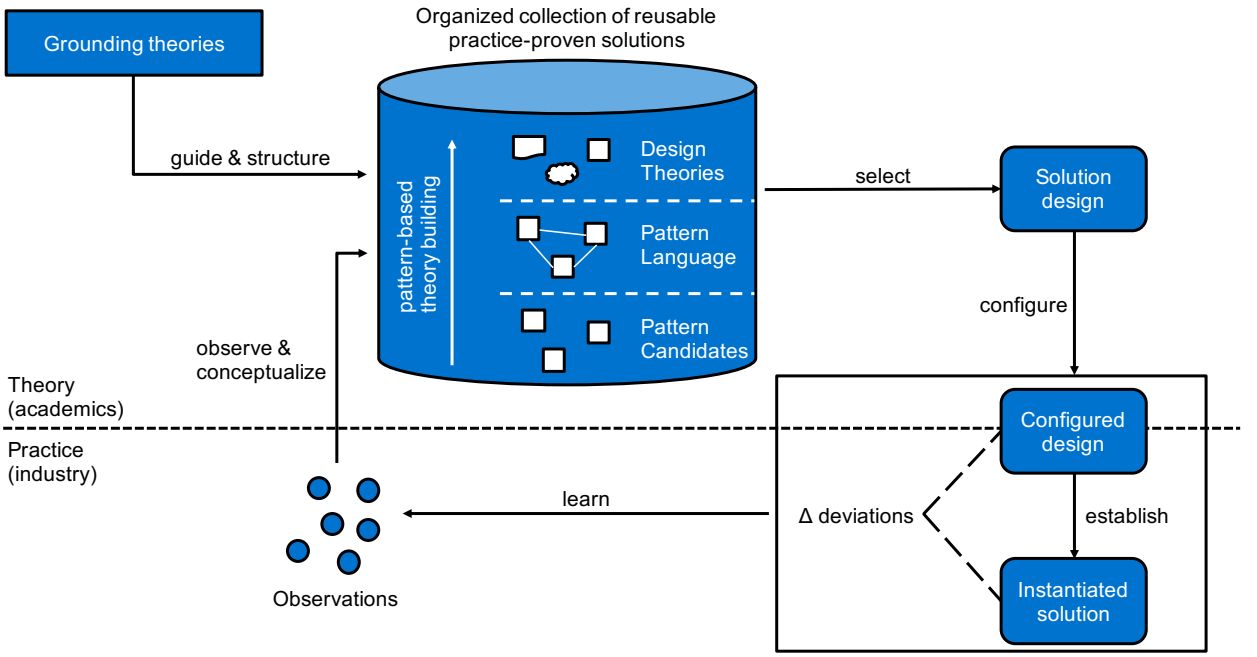

Fig. 1: Pattern-based design research [Buckl et al. 2013]

discovered four success factors, namely *unification of views and values*, *executive sponsorship and management support*, *company culture*, and *prior agile and lean experience*. In a previous study, we identified typical concerns of stakeholders and initiatives in large-scale agile development based on a structured literature review [Uludağ et al. 2018]. In total, we identified 79 concerns that were grouped into eleven challenge categories. Our previous work forms the basis for this paper, as it also included concerns and pattern candidates of agile coaches and scrum masters. [Meszaros and Doble 1997] recommend reading other related pattern languages when writing patterns. By doing that, we identified some related pattern languages as shown in Table I.

Table I. : Overview of Related Pattern Languages

Source	Scope & goal	Focus on agile development	Number of patterns	Pattern examples
[Coplien 1995]	Collection of patterns for shaping a new organization and its development processes	Partially	42	- Code Ownership - Gatekeeper - Fire Walls
[Harrison 1996]	Collection of patterns for creating effective software development teams	No	4	- Unity of Purpose - Diversity of Membership - Lock 'Em Up Together
[Beedle et al. 1999]	Collection of Scrum patterns	Yes	3	- Sprint - Backlog - Scrum Meetings
[Taylor 2000]	Collection of patterns for creating product software development environments	No	9	- Deliverables to Go - Pulse - Bootstrapping

Table I – *Continued from previous page*

Source	Scope & goal	Focus on agile development	Number of patterns	Pattern examples
[Coplien and Harrison 2004]	Collection of organizational patterns that are combined into a collection of four pattern languages	Yes	94	- SKILL MIX - DEMO PREP - FEW ROLES
[Elssamadisy 2008]	Collection of patterns for successfully adopting agile practices	Yes	38	- REFACTORING - CONTINUOUS INTEGRATION - SIMPLE DESIGN
[Beedle et al. 2010]	Collection of the most essential best practices of Scrum	Yes	11	- DAILY SCRUM - SPRINT BACKLOG - SPRINT REVIEW
[Välimäki 2011]	Enhancing performance of project management work through improved global software project management practice	Partially	18	- COLLOCATED KICK-OFF - CHOOSE ROLES IN SITES - ITERATION PLANNING
[Mitchell 2016]	Collection of patterns to address agile transformation problems	Yes	54	- LIMITED WIP - KANBAN SANDWICH - CONTROLLED FAILURE
[ScrumPLoP 2019]	Body of pattern literature around agile and Scrum communities	Yes	234 (10)	- SCRUM MASTER - SCRUM OF SCRUMS - PORTFOLIO STANDUP
[Uludağ et al. 2019]	Collection of recurring concerns and patterns of typical stakeholders in large-scale agile development	Yes	70	- STRICTLY SEPARATE BUILD AND RUN STAGES - COMMUNITY OF PRACTICE - ITERATION DEPENDENCY MATRIX - DON'T USE AGILE AS A GOLDEN HAMMER

4 Large-Scale Agile Development Pattern Language

The application of agile methods on a large scale also entails unique concerns for agile coaches and scrum masters such as establishing an agile culture & mindset across the organization, facilitating coordination and communication of multiple large-scale agile endeavors, and creating information sharing and knowledge networks [Uludağ et al. 2018; Dikert et al. 2016; Alsaqaf et al. 2019; Šmite et al. 2017]. Valuable research studies providing explanations of how to address these concerns remain still scarce.

Following the idea of [Alexander 1977], the documentation of recurring concerns and best practices of agile coaches and scrum masters seems to be useful in this context. In the following, we will present the structure of our pattern language [Uludağ et al. 2019] which forms the basis for documenting of recurring concerns and patterns of agile coaches and scrum masters (see Fig. 2).

The pattern language consists of three types of patterns [Uludağ et al. 2019]:

— **Coordination Patterns (C-Patterns)** document proven coordination mechanisms to address recurring coordination concerns, i.e., managing dependencies between activities, resources or tasks.

— **Methodology Patterns (M-Patterns)** document concrete steps to be taken to address given concerns.

— **Viewpoint Patterns (V-Patterns)** document proven ways to visualize information in the form of boards, documents, metrics, models, and reports to address recurring concerns.

In addition, the pattern language comprises the following four concepts [Uludağ et al. 2019]:

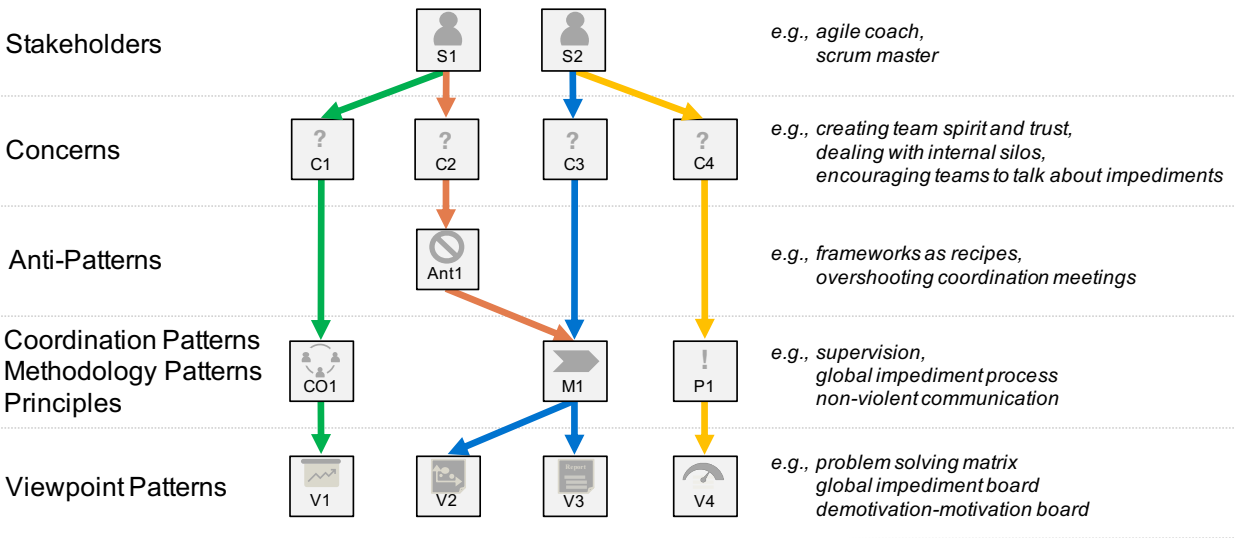

Fig. 2: Conceptual overview of the Large-Scale Agile Development Pattern Language [Uludağ et al. 2019]

— **Stakeholders** are all persons who are actively involved in, have an interest in or are in some way affected by large-scale agile development.

— **Concerns** can manifest themselves in many forms, e.g., expectations, goals, needs or responsibilities.

— **Principles** are general rules and guidelines that address given concerns by providing a common direction for action. In comparison to patterns, they do not provide any descriptions on *'how'* to address concerns.

— **Anti-Patterns (A-Patterns)** document typical mistakes and present revised solutions, which help pattern users to prevent these pitfalls.

Fig. 3 shows the attributes used to document patterns and concepts similar to those of [Buschmann et al. 1996; Ernst 2010; Coplien 1996]. All elements have an *identifier* and *name* sections to simplify referencing. Except for concerns, all elements of the pattern language have an *alias* section that contains a list of synonyms. A concern has two additional sections called *category* and *scaling level* which describe the category and the organizational level at which a concern occurs. Further, principles, patterns, and anti-patterns comprise eight common sections: the (1) *problem* and (2) *context* sections describe problems and situations to or in which they apply. The (3) *forces* section describes why the problem is tough to solve. The (4) *summary* section briefly describes the principle, pattern or anti-pattern. The (5) *consequences* section provides a list of related advantages and liabilities, while the optional (6) *other standards* and (7) *see also* sections point to other solutions and frameworks. The (8) *example* section demonstrates the problem to be addressed. Principles and patterns also have *variants* and *known uses* sections that show variants and alternatives as well as proven applications in practice. The *type* and *binding nature* sections are specific to principles and indicate their topic and whether they are recommended or mandatory. The *solution* section describes the recommended solution for a pattern. The *general form* and *revised solution* sections specific to A-Patterns delineate the not working solution and a revised solution. V-Patterns have the *type* and *data collection* sections which show the visualization concepts and collection processes necessary for their creation [Uludağ et al. 2019].
Like [Buschmann et al. 2007], we label our patterns with the star notation to denote our confidence in the maturity of a pattern. Two stars indicate that the pattern effectively solves a genuine problem in its current form. One star

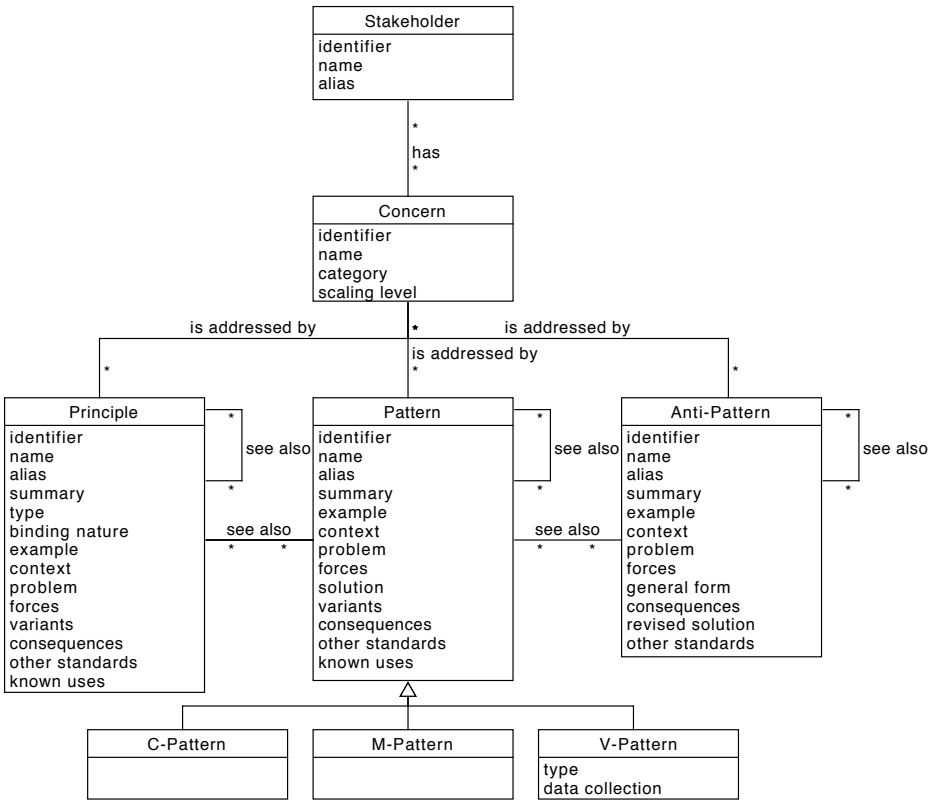

Fig. 3: Conceptual model of the Large-Scale Agile Development Pattern Language [Uludağ et al. 2019]

means that the pattern addresses a genuine problem but needs to mature. No stars denote that the pattern is a useful solution to an observed problem but requires a major revision.

5 Recurring Concerns and Best Practices

We used an integrated approach to identify recurring concerns and best practices [Cruzes and Dyba 2011]. In the first step, we created an a priori list of concerns and pattern candidates identified by a structured literature review [Uludağ et al. 2018]. In the second step, we used semi-structured interviews to prove the practical relevance of our previously identified elements as well as to extend our initial list by new concerns and best practices. All questions within the semi-structured interviews contained a combination of open and closed questions and were conversational to allow interviewees to explore their experiences and views in detail [Yin 2008]. Each interview was primarily conducted by two researchers in face-to-face meetings to facilitate observer triangulation [Runeson and Höst 2009]. A total of 13 interviews were conducted with agile coaches and scrum masters (see Table II).
In total, we observed 57 recurring concerns of agile coaches and scrum master, 36 of which were already identified by the literature review [Uludağ et al. 2018] and 21 of which were newly mentioned by the interviewees. A detailed list of identified concerns can be found in Appendix A.
We identified a total of 76 pattern candidates consisting of 21 M-Patterns, 18 V-Patterns, 14 C-Patterns, 12 Principles, and 10 A-Patterns as shown in Fig. 4. After applying the rule of three [Coplien 1996], we identified a total of 15 patterns comprising 5 M-Patterns, 2 V-Patterns, 2 C-Patterns, 4 Principles, and 2 A-Patterns. A detailed list of identified patterns can be found in Appendix B.

Table II. : Overview of interview partners

No	Role	Professional experience (in years)	Organization's experience (in years)	Industry
1	Agile Coach	3-6 years	> 6 years	IT / Technology
2	Agile Coach	3-6 years	1-3 years	Production
3	Agile Coach	1-3 years	3-6 years	Consulting
4	Agile Coach	> 6 years	1-3 years	IT / Technology
5	Agile Coach	3-6 years	< 1 year	Consulting
6	Agile Coach / Scrum Master	> 6 years	3-6 years	IT / Technology
7	Agile Coach	3-6 years	3-6 years	Consulting
8	Agile Coach	1-3 years	1-3 years	Finance / Insurance / Real Estate
9	Agile Coach	1-3 years	1-3 years	Retail
10	Agile Coach	> 6 years	1-3 years	Production
11	Agile Coach	3-6 years	1-3 years	Consulting
12	Agile Coach	3-6 years	3-6 years	Retail
13	Agile Coach	3-6 years	3-6 years	Consulting

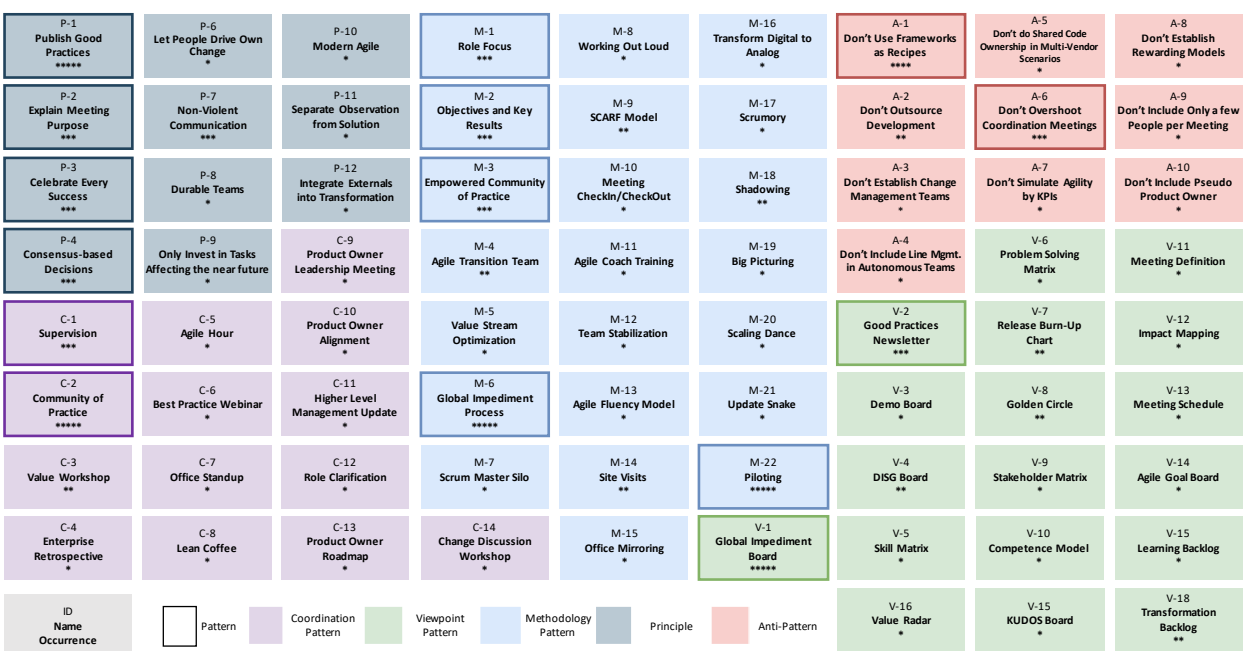

Fig. 4: Overview of identified patterns and pattern candidates

Fig. 5 depicts the current version of our pattern language, which visualizes the relationships[2] between recurring concerns and patterns of agile coaches and scrum masters. Hereafter, we present five best practices that showcase

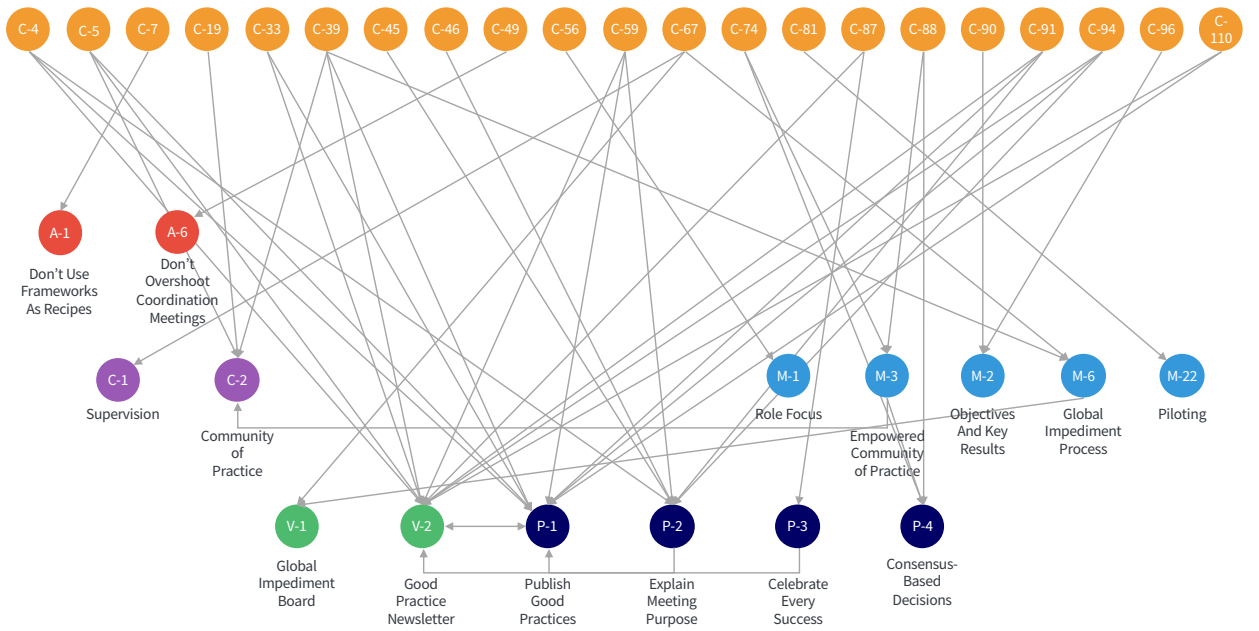

Fig. 5: Pattern language for agile coaches and scrum masters *

the different pattern types and concepts of our pattern language:

(1) **P-1:** PUBLISH GOOD PRACTICES (showing Principles),
(2) **C-1:** SUPERVISION (representing C-Patterns),
(3) **M-6:** GLOBAL IMPEDIMENT PROCESS (highlighting M-Patterns),
(4) **V-6:** GLOBAL IMPEDIMENT BOARD (illustrating (V-Patterns), and
(5) **A-1:** DON'T USE SCALING AGILE FRAMEWORKS AS A RECIPE (demonstrating A-Patterns).

[2]The arrows between the orange and other circles indicate the *is addressed by* relationship between concerns and pattern types/concepts. For instance, the concern **C-19:** *How to deal with internal silos? is addressed by* the C-Pattern **C-2:** COMMUNITY OF PRACTICE. The relationships between pattern types and/or concepts represent the *uses* or the *can be used in combination* relationship (cf. [Buschmann et al. 1996]). For example, the M-Pattern **M-2:** GLOBAL IMPEDIMENT PROCESS can be used in combination with the V-Pattern **V-1:** GLOBAL IMPEDIMENT BOARD to address the concern **C-67:** *How to encourage development teams to talk about tasks and impediments?*.

5.1 Principle: Publish Good Practices (P-1) *

Principle Overview	
Alias	
Summary	PUBLISH GOOD PRACTICES to enable a culture of open communication and continuous improvement. By applying PUBLISH GOOD PRACTICES, agile teams are encouraged to talk about things that what went well and to share their achievements with other agile teams.
Type	Agile Principle
Binding Nature	Recommended

5.1.1 *Example*

A scrum master at RetailCo successfully built a culture of kudos within his team. Although the team's motivation was highly increased by the kudos-giving culture, the scrum master did not share his success story with other scrum masters due to missing communication channels for organizational learning as well as and due to the lack of a continuous improvement culture at RetailCo.

5.1.2 *Context*

In the course of large-scale agile transformations, agile teams learn new practices and quickly achieve remarkable achievements in their new way of working. Although their achievements can also be of great importance to other agile teams, their success stories are not further communicated and the valuable knowledge remains inaccessible to the organization.

5.1.3 *Problem*

The following concerns are addressed by PUBLISH GOOD PRACTICES:

— **C-4:** *How to deal with doubts in people about changes?*
— **C-5:** *How to facilitate shared context and knowledge?*
— **C-33:** *How to build trust of stakeholders in agile practices?*
— **C-39:** *How to establish a culture of continuous improvement?*
— **C-59:** *How to establish a common understanding of agile thinking and practices?*
— **C-91:** *How to demonstrate the value add of agile methods?*
— **C-94:** *How to understand the demand for becoming agile?*
— **C-110:** *How to establish an agile mindset?*

5.1.4 *Forces*

The following forces influence PUBLISH GOOD PRACTICES:

— Agile teams are not conscious about the importance of sharing their achievements with other agile teams.
— There are no communication channels to share good practices across the organization.

5.1.5 *Consequences*

The following benefits of PUBLISH GOOD PRACTICES are known:

— A culture of continuous learning is established.
— Open communication is facilitated.
— People get informed about good practices within and outside their organization.
— People are engaged to try out new things.

The following liabilities of PUBLISH GOOD PRACTICES are known:

— Applying this principle can lead to a higher tolerance to make mistakes and trying out things that are not suitable for the organization or the team.
— An excessive use of this principle could make it difficult for agile teams to distinguish between important and unimportant good practices and to identify practices that are relevant to them.

5.1.6 *See Also*

Publish Good Practices can be used in combination with the following V-Pattern:

— **V-2:** GOOD PRACTICE NEWSLETTER

5.1.7 *Known Uses*

The following uses of PUBLISH GOOD PRACTICES are known:

— AgileConsultCo
— CarCo
— InsureCo
— ITConsultCo
— RetailCo

5.2 C-Pattern: Supervision (C-1) **

C-Pattern Overview	
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.

5.2.1 *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.

5.2.2 *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.

5.2.3 *Problem*

The following concern is addressed by SUPERVISION:

— **C-67:** *How to encourage development teams to talk about tasks and impediments?*

5.2.4 *Forces*

The following forces influence SUPERVISION:

— 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.

5.2.5 *Solution*

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.

5.2.6 *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.

5.2.7 *Consequences*

The following benefits of SUPERVISION are known:

— 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 of SUPERVISION are known:

— 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.

5.2.8 *Known Uses*

The following uses of SUPERVISION are known:

— ConglomerateCo
— RetailCo
— SoftwareConsultCo

5.3 M-Pattern: Global Impediment Process (M-6) *

M-Pattern Overview	
Alias	
Summary	The GLOBAL IMPEDIMENT PROCESS describes a structured process for identifying, documenting, and solving impediments that affect multiple agile teams.

5.3.1 *Example*

A large-scale agile development program of RetailCo consisting of seven agile teams are about to finish their first two-week sprint. The scrum masters of these teams request access rights from the infrastructure team to use the testing environment. However, the infrastructure team is not able to process these requests since all virtual machines are already used by other teams. Consequently, the first sprint of the agile teams cannot be completed because they could not test the software sufficiently. In the respective team retrospectives, this impediment is raised by the developers. Since this is impediment is a big issue at RetailCo, the scrum masters are not able to solve it themselves. Also, RetailCo does not have a pre-defined process for escalating this impediment at the enterprise level.

5.3.2 *Context*

In agile software development, scrum masters are primarily responsible for removing impediments that may slow the development progress of their respective teams. However, in complex software development endeavors, in which multiple agile teams are involved in, these impediments are more difficult to solve as not only one agile team is affected by it but multiples. Since large-scale agile development endeavors also include other organizational units in the development process, the scrum masters do not have sufficient instruction authorities to induce other employees outside of their teams to perform specific tasks for resolving impediments of their teams.

5.3.3 *Problem*

The following concerns are addressed by GLOBAL IMPEDIMENT PROCESS:

— **C-39:** *How to establish a culture of continuous improvement?*

— **C-67:** *How to encourage development teams to talk about tasks and impediments?*

5.3.4 *Forces*

The following forces influence GLOBAL IMPEDIMENT PROCESS:

— Impediments in large-scale agile development typically affect multiple agile teams.

— Scrum masters lack mechanisms to escalate larger impediments to higher organizational levels so that these are resolved by middle management or even by the executive board.

— Scrum masters do not have sufficient instruction authorities of employees outside of their teams that should take actions in order to resolve the impediments.

— Scrum masters have difficulties in identifying relevant employees outside of their teams that should own and resolve the impediments.

5.3.5 *Solution*

Implement a GLOBAL IMPEDIMENT PROCESS to tackle impediments that scrum masters cannot solve on their own. Each impediment of a team is included within the GLOBAL IMPEDIMENT PROCESS if the team cannot solve the impediment by its own. The process is structured according to Fig. 6.

The process includes a *Global Impediment Working Group* that consists of people having an overarching view of the organization, therefore knowing the right people to solve a global impediment. The Global Impediment Working Group meets every two weeks and discusses and prioritizes new impediments. They add them to the

Global Impediment Board and try to solve the impediments. Because of their knowledge about the company, the probability that they know people who can solve the impediment is very high.

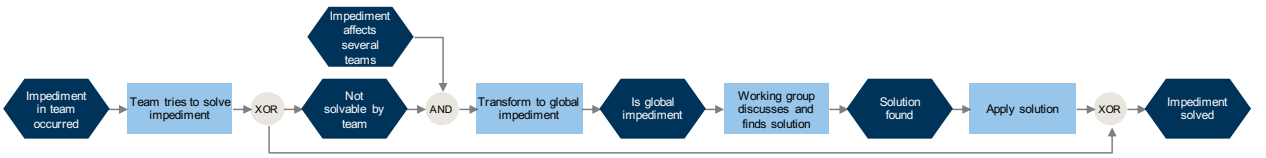

Fig. 6: Event-Driven GLOBAL IMPEDIMENT PROCESS

5.3.6 *Variants*

The following variants are known for the GLOBAL IMPEDIMENT PROCESS:

— Each impediment has an 'owner', who is someone from the Working Group who also knows about the difficulty of the impediment.
— Each impediment has a 'supporter', who is someone who as a major influence on the solution of the impediment.
— An impediment can be documented by using the A3 format [Sobek and Jimmerson 2004].

5.3.7 *Consequences*

The following benefits of GLOBAL IMPEDIMENT PROCESS are known:

— Impediments get solved.
— Prioritization enables calculation of real cost caused by an impediment. This may increase resolution speed.
— The process enables transparency.
— The process minimized local applications and workarounds.
— By resolved impediments, the velocity of agile teams is not hampered.

The following liabilities of GLOBAL IMPEDIMENT PROCESS are known:

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

5.3.8 *See Also*

The GLOBAL IMPEDIMENT PROCESS uses the following V-Pattern:

— **V-1:** GLOBAL IMPEDIMENT BOARD

5.3.9 *Known Uses*

The following uses of GLOBAL IMPEDIMENT PROCESS are known:

— AutonomousDrivingCo
— ConglomerateCo
— RetailCo
— SoftwareCo
— SoftwareConsultCo

5.4 V-Pattern: Global Impediment Board (V-6)*

V-Pattern Overview	
Alias	Global Impediment Backlog
Summary	A GLOBAL IMPEDIMENT BOARD shows all impediments of an organization which are ether not solvable by an agile team itself or are relevant for several teams in a company.
Type	Board

5.4.1 *Example*

RetailCo has established a GLOBAL IMPEDIMENT PROCESS to handle impediments that scrum masters cannot solve on their own or are relevant for multiple agile teams. Although RetailCo has established a Global Impediment Working Group for resolving these type of impediments, it neither uses a structured format for documenting global impediments nor stores them in a central database. In addition, the Global Impediment Working Group does not prioritize global impediments. Thus, it does not directly addressed urgent impediments that can have a significant impact on agile teams.

5.4.2 *Context*

The organization has already implemented the M-Pattern GLOBAL IMPEDIMENT PROCESS. The resolution of impediments is neither documented in a uniform format nor stored centrally so that employees have difficulties in identifying current or historical global impediments. In addition, the organization does not know which of the impediments are important or urgent to resolve.

5.4.3 *Problem*

The following concern is addressed by GLOBAL IMPEDIMENT BOARD:

— **C-67:** *How to encourage development teams to talk about tasks and impediments?*

5.4.4 *Forces*

The following forces influence GLOBAL IMPEDIMENT BOARD:

— Global Impediments need to be centrally managed and tracked so that they are actually resolved by the impediment owners.

— Different employees may have different styles of documenting impediments that would make it difficult to compare impediments.

— Due to numerous tools in the area of large-scale agile development, it can be difficult to find global impediments in the right place.

— Due to vast numbers of global impediments, it can be difficult to distinguish important/urgent impediments from unimportant/non-urgent impediments.

5.4.5 *Solution*

Set up a GLOBAL IMPEDIMENT BOARD to manage all global impediments throughout the GLOBAL IMPEDIMENT PROCESS. In large-scale agile development, a list with the following structure is frequently used:
The ID is a consecutive, unique integer value that is used to identify an impediment. The prioritization is done by the Global Impediment Working Group and indicates the urgency of the impediment. Impediments with higher prioritization should be solved first. 'Handed in by' refers to the team or individual who handed in the impediment. The owner is someone from the Working Group, who is responsible for solving the impediment. The 'A3'-attribute is optional if the GLOBAL IMPEDIMENT PROCESS requires the submission of an impediment in the A3 format.
The underlying information model of the GLOBAL IMPEDIMENT BOARD can be found in Fig. 8.

Global Impediment Board

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

Fig. 7: Exemplary view for GLOBAL IMPEDIMENT BOARD

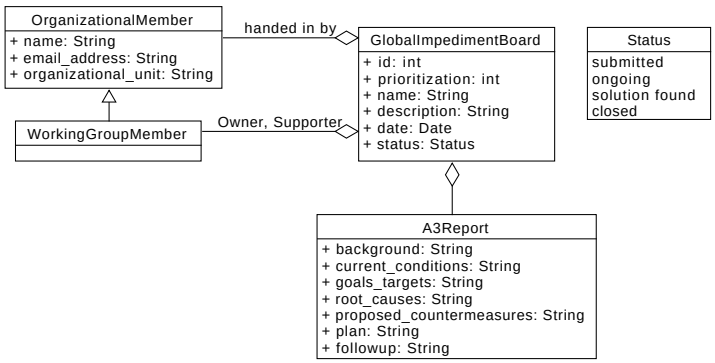

Fig. 8: Underlying information model of GLOBAL IMPEDIMENT BOARD

5.4.6 *Variants*

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

5.4.7 *Consequences*

The following benefits of GLOBAL IMPEDIMENT BOARD are known:

— Everyone within the organization can view current global impediments and see if anyone else has a similar problem.

— Prioritization enables faster solving of emerging impediments that have a large impact on a team's organization.

— Global impediments can be compared more easily with each other.

The following liabilities of GLOBAL IMPEDIMENT BOARD are known:

— It requires effort to manage the GLOBAL IMPEDIMENT BOARD.

5.4.8 *Data Collection*

The GLOBAL IMPEDIMENT BOARD can be digitally administrated in any digital collaboration tool by the Global Impediment Working Group. It is updated whenever an impediment occurs by a member of the Working Group. Only these members have writing permissions. It depends on the organization if the board should be public or kept private to the Working Group.

5.4.9 *See Also*

GLOBAL IMPEDIMENT BOARD is used by the following M-Pattern:

— **M-8:** GLOBAL IMPEDIMENT PROCESS

5.4.10 *Known Uses*

The following uses of GLOBAL IMPEDIMENT BOARD are known:

— AutonomousDrivingCo
— ConglomerateCo
— RetailCo
— SoftwareCo
— SoftwareConsultCo

5.5 A-Pattern: Don't Use Scaling Agile Frameworks as a Recipe (A-1) *

A-Pattern Overview	
Alias	Scaling Agile Frameworks Aren't Silver Bullets
Summary	The A-Pattern DON'T USE SCALING AGILE FRAMEWORKS AS A RECIPE shows why it is not advisable to over rely on scaling agile frameworks without training people in agile values and principles and tailoring these frameworks to the specific needs of the organization.

5.5.1 *Example*

The executive board of RetailCo decides to revive a failed customer relationship project by using agile methods. Due to the complexity of the project, the management decide to relaunch it with the help of a scaling framework. After a short internet research and discussion with external agile coaches, the management decides to relaunch the project by adopting the Scaled Agile Framework[3] (SAFe). In order not to fail, the management decides to use the most comprehensive configuration of the SAFe framework, namely the Full SAFe configuration. After hiring several external agile coaches and offering one-day SAFe training courses for the employees, the large-scale agile development program consisting with three agile teams starts with the first program increment. The management of RetailCo stipulated the large-scale agile development program that all practices, artefacts and roles specified by Full SAFe are to be applied. After a few program increments the large-scale agile development program notices that the used framework is causing some problems such as additional workload or complexity. Consequently, the large-scale agile development program compulsively tries to solve the problems arising from the use of Full SAFe and neglects to address the actual problems of the project.

5.5.2 *Context*

As today's competitive environments become increasingly turbulent and the software systems to be developed become more complex, traditional companies increasingly decide to adopt scaling agile frameworks for their software development endeavors. Thus, several scaling agile frameworks, such as DAD[4], LeSS[5], and SAFe[6] were proposed by practitioners to resolve issues associated with team size, customer involvement, and project constraints that are mostly applied by traditional organizations.

5.5.3 *Problem*

The following concern is addressed by DON'T USE SCALING AGILE FRAMEWORKS AS A RECIPE:

— **C-7:** *How to deal with incorrect practices of agile development?*

5.5.4 *Forces*

The following forces occur in the context of DON'T USE SCALING AGILE FRAMEWORKS AS A RECIPE:

— The vendors of scaling agile frameworks promise organizations that the full potential of scaling agile frameworks will only be achieved when companies implement the frameworks exactly as they require.

— Companies are tempted by the fact that the frameworks have already been successfully implemented in other organizations and therefore automatically assume that these will also solve current problems in their organization.

— Due to the lack of an agile mindset, companies believe that the mere use of scaling agile frameworks is sufficient to realize the benefits of agile development, neglecting the importance of agile principles and values.

[3] https://www.scaledagileframework.com/
[4] https://disciplinedagiledelivery.com/
[5] https://less.works/
[6] https://www.scaledagileframework.com/

— Due to their previous way of thinking, traditional organizations tend to think in terms of predefined templates and processes, thus limiting the ability to discover new ideas and ways of working.

5.5.5 *General Form*

Traditional organizations that worked for a long period in hierarchical structures and plan-driven software development methods tend to over-rely on predefined structures, processes, and rules. However, this mindset is also exercised when organizations decide to adopt scaling agile frameworks as a basis for their complex software development endeavors. Thus, these type of organizations adopt the practices, artefacts, and roles proposed by the frameworks without to question whether they are useful for the addressing the actual problems of the organization. In this context, organizations focus more on the appropriate implementation of the adopted scaling agile framework than on understanding the values and principles behind the framework. This phenomenon is also known as *'method prison'*.

5.5.6 *Consequences*

The following benefits of DON'T USE SCALING AGILE FRAMEWORKS AS A RECIPE are known:

— Scaling agile frameworks provide detailed guidance for applying agile practices.
— Scaling agile frameworks constitute an entry point for hierarchical organizations to establish an agile culture across the company.
— The usage of scaling agile frameworks increases the productivity of the organization.
— Scaling agile frameworks provide quick answers for typical software process problems.

The following liabilities of DON'T USE SCALING AGILE FRAMEWORKS AS A RECIPE are known:

— By relying too much on scaling agile frameworks, employees are not encouraged to understand the values and principles behind these frameworks and do not develop an agile mindset.
— Not all practices, roles, and artefacts of scaling agile frameworks apply to an enterprise, so wasting time, money, and effort is put into their application.
— Since scaling agile frameworks represent a simplified representation of reality, they are not designed to address more complex problems from reality.

5.5.7 *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.

5.5.8 *See Also*

The A-Pattern DON'T USE SCALING AGILE FRAMEWORKS AS A RECIPE can be avoided by using the following principle:

— **P-17:** SEPARATE OBSERVATION FROM SOLUTION

6 Discussion

In the following, we discuss the main findings of our study.

(1) Adaptation of scaling agile frameworks to company contexts
The majority of the interviewed organizations adopted a variety of scaling agile frameworks for supporting their product development such as Large-Scale Scrum[7], Scaled Agile Framework[8], Scrum of Scrums[9], and Scrum at Scale[10]. Some of the interviewed companies invented their frameworks mainly for scaling agile practices over multiple teams. In larger companies, we observed that various scaling agile frameworks were used in different product development units. Furthermore, we noticed that the importance of the frameworks for the companies differed significantly. While in some organizations the correct implementation of a certain framework was regarded as very important for the success of the product development, other companies considered scaling agile frameworks as a means to an end. Typically, the latter type of organizations used the frameworks as inspiration for their product development. Organizations concentrating too much on the proper implementation of a specific framework tended to fall into the pitfall of using frameworks as recipes (see A-Pattern DON'T USE SCALING AGILE FRAMEWORKS AS A RECIPE). Although many framework vendors strongly recommend the complete and unmodified use of their frameworks, the interviewees mentioned that their organizations have not adopted the scaling of agile frameworks one-to-one, but have tailored them to their context and needs. These adaptations included: (1) renaming of roles, practices, and artifacts, (2) introducing new roles based on the current organizational unit, (3) initiating new coordination meetings due to increased coordination needs, and (4) omitting recommended roles, events, and artifacts in order not to further increase the complexity of product development.

(2) Risk of patterns being used as cooking recipes
Our decision to document best practices in the form of patterns received positive feedback by the agile and scrum masters as well as by the interviewees from our previous study (cf. [Uludağ et al. 2019]). On the one hand, the interviewees asked themselves how these patterns could be used pedagogically to train new employees in the field of large-scale agile development. On the other hand, other participants considered using the patterns for upcoming projects. Similar to scaling agile frameworks, the usage of patterns also harbors some risks that must be mentioned here. First, patterns are context-specific signifying that some patterns may work very well in some companies, while the same patterns may not be suitable for other organizations. For instance, while some organizations preferred to introduce change teams (see M-Pattern AGILE TRANSITION TEAM) aiming to lead the agile transformations within the organizations, others stated that establishing change teams might not work, and thus should be avoided (see A-Pattern DON'T ESTABLISH CHANGE MANAGEMENT TEAMS). One way to mitigate this risk is by applying the PDR method presented in Section 2. According to the PDR method, a researcher should support organizations in the selection of suitable patterns and their configuration for the organizational context. In addition, the researcher should document possible deviations and, if necessary, revise the initial pattern, e.g., by updating the consequences or variants sections of a pattern. Second, similar to the application of scaling agile frameworks, employees may focus excessively on the correct application of a pattern rather than understanding the actual problem and intentions behind the pattern. Thus, we highly recommend that patterns should be used as decision-support and should not anticipate decisions.

(3) Agile values and principles vs. agile practices
During the interviews, many agile coaches and scrum master pointed out that many employees wrongly equate

[7] https://less.works/less/framework/index.html
[8] https://www.scaledagileframework.com/
[9] https://www.scruminc.com/scrum-of-scrums/
[10] https://www.scrumatscale.com/scrum-at-scale-guide/

agility by using agile practices without understanding the underlying values and principles, e.g., moving cards on Kanban boards or replacing *'old and boring'* software development terminologies with *'new and fancy'* agile development terminologies without knowing *'why'* these practices are important. As a consequence, many companies become *'pseudo agile'* which increasingly encounter cultural problems, since the agile mindset of the employees is still very immature. Also, the interviewed agile coaches and scrum masters indicated that learning new practices is much easier than understanding the underlying values and principles. They explained this problem by using the metaphor of an iceberg, i.e., the visible part of an iceberg represents agile practices and the important part of the iceberg, which is underwater, is made of agile principles and values. Influencing values and principles which are not visible is more difficult than influencing the visible practices.

(4) Process-oriented agile coaches vs. mindset-oriented agile coaches
We observed two types of agile coaches in our interviews, namely *process-oriented* and *mindset-oriented* agile coaches. The process-focused coaches were mainly concerned about the proper application of agile practices and methods by the teams. To this end, they mainly proposed best practices in the form of M-Patterns such as OBJECTIVES AND KEY RESULTS and GLOBAL IMPEDIMENT PROCESS. On the other hand, the mindset-focused coaches often explained concerns about the adoption of an agile mindset across the organization or the creation of an ideal working environment for agile teams to foster team communication and collaboration. The mindset-oriented coaches named typically principles, workshops, and visualizations for resolving their concerns, such as the Principle PUBLISH GOOD PRACTICES and the C-Pattern SUPERVISION. These observations are consistent with the concept of [Kelly 2008], according to which there are two different approaches to coaching: *directive* and *non-directive* coaching. The process-focused coaches often applied a directive approach, while the mindset-focused coaches mostly used a non-directive coaching approach. With directive coaching, the agile coach has extensive knowledge of the domain and mostly trains a team in the application of agile practices. Thus, directive coaching can be used to adopt agile practices and help teams to work in new ways. In contrast, the non-directive approach does not necessarily require the coach to be an expert in the field. Instead, the coach tries to help the teams focus on their own goals and work towards achieving them. This form of coaching helps the teams to grow on their own and to improve their performance. Non-directive coaching is more suitable for teams that are already familiar with agile practices, while the directive approach is more suited for new teams [Kelly 2008].

7 Conclusion and Outlook

The success of agile methods for small teams has inspired large enterprises to apply them on a larger scale to build complex software systems [Dingsøyr and Moe 2014; Alqudah and Razali 2016]. The scaling of agile methods entails key managerial challenges such as coordinating multiple large-scale agile endeavors, establishing an agile mindset across the organization, and facing general resistances to changes [Uludağ et al. 2018; Dikert et al. 2016; Alsaqaf et al. 2019]. Especially agile coaches and scrum master are confronted with a number of unprecedented concerns in large-scale agile development [Uludağ et al. 2018]. Notwithstanding the significance of agile coaches and scrum masters for the success of large-scale agile endeavors, extant literature disregards an overview of their concerns and a collection of best practices to address them. Against this backdrop, we interviewed 13 agile coaches and scrum masters and identified 57 recurring concerns and 15 best practices, five of which were presented in this paper.
Finally, this paper leaves some room for future research. First, we aim to conduct more interviews with other typical stakeholders in large-scale agile development, such as product owners, solution architects, and developers. These interviews will help us to identify new role-specific concerns, patterns, and pattern candidates. Second, by means of a structured questionnaire among companies worldwide, we will publish the Large-Scale Agile Development Pattern Catalog containing concerns and patterns. Third, we will assist agile coaches and scrum masters of our industry partners in selecting relevant patterns and introducing them into their organizations. This will allow us to

observe actual pattern instantiations and to identify possible deviations from the originally introduced patterns. Thereby, we also intend to close the research activity cycle of the PDR method [Buckl et al. 2013].

Acknowledgements

This work has been sponsored by the *German Federal Ministry of Education and Research (BMBF)* via the Software Campus Project SaM-IT 01IS17049 project.

REFERENCES

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

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

Wasim Alsaqaf, Maya Daneva, and Roel Wieringa. 2019. Quality requirements challenges in the context of large-scale distributed agile: An empirical study. *Information and Software Technology* 110 (2019), 39 – 55.

Kent Beck. 2000. *Extreme programming explained: embrace change*. Addison-Wesley.

Mike Beedle, James O. Coplien, Jeff Sutherland, Jens C. Østergaard, Ademar Aguiar, and Ken Schwaber. 2010. Essential Scrum Patterns. In *14th European Conference on Pattern Languages of Programs*. The Hillside Group, Irsee, 1–17.

Mike Beedle, Martine Devos, Yonat Sharon, Ken Schwaber, and Jeff Sutherland. 1999. SCRUM: An Extension Pattern Language for Hyperproductive Software Development. *Pattern Languages of Program Design* 4 (1999), 637–651.

Sabine Buckl, Florian Matthes, Alexander W. Schneider, and Christian M. Schweda. 2013. Pattern-Based Design Research – An Iterative Research Method Balancing Rigor and Relevance. In *8th International Conference on Design Science Research in Information Systems*. Springer, Berlin, 73–87.

Frank Buschmann, Kevlin Henney, and C. Schmidt Douglas. 2007. *Pattern Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing*. John Wiley & Sons, Chichester.

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 1996. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. John Wiley & Sons, Chichester.

James O. Coplien. 1995. A Generative Development-process Pattern Language. In *Pattern Languages of Program Design*, James O. Coplien and Douglas C. Schmidt (Eds.). ACM, New York, 183–237.

James O. Coplien. 1996. *Software Patterns: Management Briefs*. Cambridge university Press, Cambridge.

James O. Coplien and Neil B. Harrison. 2004. *Organizational Patterns of Agile Software Development*. Addison-Wesley, Boston.

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

Kim Dikert, Maria Paasivaara, and Casper Lassenius. 2016. Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review. *Journal of Systems and Software* 119 (2016), 87–108.

Torgeir Dingsøyr and Nils Moe. 2014. *Towards Principles of Large-Scale Agile Development*. Springer, Berlin, 1–8.

Amr Elssamadisy. 2008. *Agile Adoption Patterns: A Roadmap to Organizational Success*. Addison-Wesley, Boston.

Alexander M. Ernst. 2010. *A Pattern-based Approach to Enterprise Architecture Management*. Dissertation. Technische Universität München, München.

Neil B. Harrison. 1996. Organizational Patterns for Teams. In *Pattern Languages of Program Design 2*, John M. Vlissides, James O. Coplien, and Norman L. Kerth (Eds.). Addison-Wesley, Boston, 345–352.

Martin Kalenda, Petr Hyna, and Bruno Rossi. 2018. Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process* 30, 10 (2018), e1954. DOI:https://doi.org/10.1002/smr.1954

Allan Kelly. 2008. *Changing software development: Learning to become agile*. John Wiley & Sons.

Petri Kettunen. 2007. Extending Software Project Agility with new Product Development Enterprise Agility. *Software Process: Improvement and Practice* 12, 6 (2007), 541–548.

Gerard Meszaros and Jim Doble. 1997. A Pattern Language for Pattern Writing. In *Pattern Languages of Program Design 3*, Robert C. Martin, Dirk Riehle, and Frank Buschmann (Eds.). Addison-Wesley, Boston, 529–574.

Ian Mitchell. 2016. *Agile Development in Practice*. TamaRe House, London.

Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2009), 131.

Ken Schwaber and Mike Beedle. 2001. *Agile Software Development with Scrum* (1st ed.). Prentice Hall, Upper Saddle River.

ScrumPLoP. 2019. Published Patterns. https://sites.google.com/a/scrumplop.org/published-patterns/. (2019). Accessed: 2019-02-02.

Darja Šmite, Nils Brede Moe, Aivars Šāblis, and Claes Wohlin. 2017. Software teams and their knowledge networks in large-scale software development. *Information and Software Technology* 86 (2017), 71–86.

Durward K. Sobek and Cindy Leduc Jimmerson. 2004. A 3 Reports : Tool for Organizational Transformation. *Industrial Engineering Research Conference*.

Paul Taylor. 2000. Capable, productive, and satisfied: Some organizational patterns for protecting productive people. In *Pattern Languages of Program Design 4*, John M. Vlissides, James O. Coplien, and Norman L. Kerth (Eds.). Addison-Wesley, Boston, 611–636.

Ömer Uludağ, Nina Harders, and Florian Matthes. 2019. Documenting Recurring Concerns and Patterns in Large-Scale Agile Development. In *24th European Conference on Pattern Languages of Programs*. ACM, New York.

Ömer Uludağ, Martin Kleehaus, Christoph Caprano, and Florian Matthes. 2018. Identifying and Structuring Challenges in Large-Scale Agile Development Based on a Structured Literature Review. In *22nd International Enterprise Distributed Object Computing Conference*. IEEE, Stockholm, 191–197.

Antti Välimäki. 2011. *Pattern Language for Project Management in Global Software Development*. Tampere University of Technology, Tampere.

VersionOne. 2018. *12th Annual State of Agile Report*. Technical Report. VersionOne.

Robert K. Yin. 2008. *Case Study Research: Design and Methods*. Sage Publications, London.

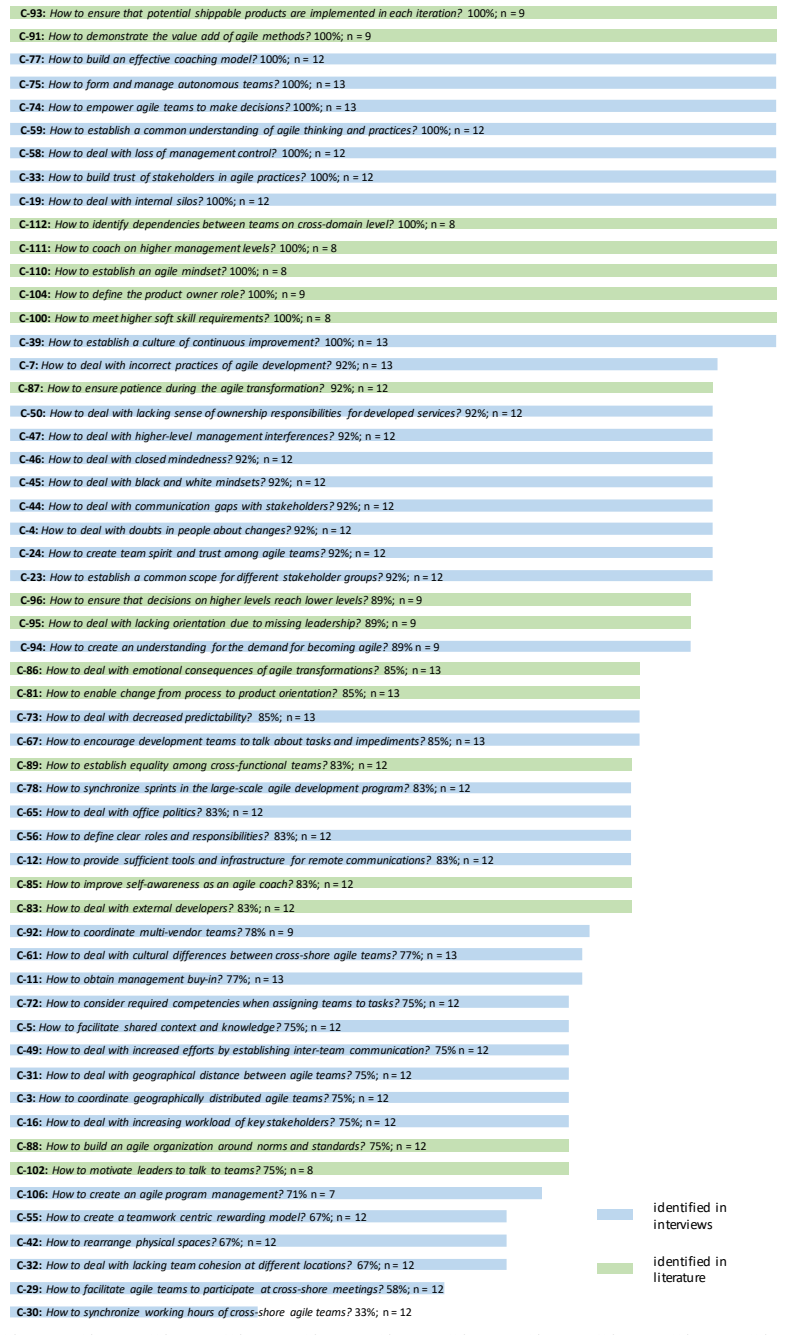

Fig. 9: Overview of identified concerns of agile coaches and scrum masters

B Pattern Language for Agile Coaches and Scrum Masters *

B.1 Concerns

— **C-4:** *How to deal with doubts in people about changes?*
— **C-5:** *How to facilitate shared context and knowledge?*
— **C-19:** *How to deal with internal silos?*
— **C-33:** *How to build trust of stakeholders in agile practices?*
— **C-39:** *How to establish a culture of continuous improvement?*
— **C-46:** *How to deal with closed mindedness?*
— **C-49:** *How to deal with increased efforts by establishing inter-team communication?*
— **C-56:** *How to define clear roles and responsibilities?*
— **C-59:** *How to establish a common understanding of agile thinking and practices?*
— **C-67:** *How to encourage development teams to talk about tasks and impediments?*
— **C-74:** *How to empower agile teams to make decisions?*
— **C-81:** *How to enable the change from process to product orientation?*
— **C-87:** *How to ensure patience during the agile transformation?*
— **C-88:** *How to build an agile organization around norms and standards?*
— **C-90:** *How to deal with the lack of objective measuring methods?*
— **C-91:** *How to demonstrate the value add of agile methods?*
— **C-94:** *How to understand the demand for becoming agile?*
— **C-96:** *How to ensure that decisions on higher levels reach lower levels?*
— **C-110:** *How to establish an agile mindset?*

B.2 A-Patterns

— **A-1:** DON'T USE FRAMEWORKS AS RECIPES
— **A-6:** DON'T OVERSHOOT COORDINATION MEETINGS

B.3 Principles

— **P-1:** PUBLISH GOOD PRACTICES
— **P-2:** EXPLAIN MEETING PURPOSE
— **P-3:** CELEBRATE EVERY SUCCESS
— **P-4:** CONSENSUS-BASED DECISIONS

B.4 M-Patterns

— **M-1:** ROLE FOCUS
— **M-2:** OBJECTIVES AND KEY RESULTS
— **M-3:** EMPOWERED COMMUNITY OF PRACTICE
— **M-6:** GLOBAL IMPEDIMENT PROCESS
— **M-22:** PILOTING

B.5 C-Patterns

— **C-1:** SUPERVISION
— **C-2:** COMMUNITY OF PRACTICE

B.6 V-Patterns

— **V-1:** GLOBAL IMPEDIMENT BOARD
— **V-2:** GOOD PRACTICE NEWSLETTER

PLoP'19, OCTOBER 7–10, Ottawa, Ontario, Canada. Copyright 2015 is held by the author(s). HILLSIDE 978-1-941652-03-9