



DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

**Participatory establishment of guidelines
through automated testing and gamification
in large-scale agile software development**

Sascha Nägele





DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

**Participatory establishment of guidelines through
automated testing and gamification in large-scale
agile software development**

**Partizipative Etablierung von Richtlinien durch
Gamifizierung und automatisiertes Testen in der
großen agilen Softwareentwicklung**

Author: Sascha Nägele
Supervisor: Prof. Dr. Florian Matthes
Advisor: Ömer Uludağ, M. Sc.
Submission Date: 15.12.2018



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

Munich, 15.12.2018

Sascha Nägele

Acknowledgments

First and foremost, I would like to thank my thesis advisor Ömer Uludağ for his great support over the course of this thesis. Your ongoing guidance, constructive feedback and enthusiasm were of great value.

Furthermore, I would also like to thank Dr. Matheus Hauder who advised me on the practical side of the thesis and who ensured the practical relevance of this thesis through his valuable support and feedback. Thank you for the exciting opportunity to write this thesis with an industry partner and the resulting interesting experiences.

In addition, I would like to thank Dr. Thomas Kofler for the close collaboration at the industry partner as well as all my other colleagues from the industry partner for participating in countless interviews and conversations, and who referred me to further relevant contacts and thus contributed a great deal to make this research possible.

I would also like to thank Professor Dr. Florian Matthes, who made this research possible in the first place by providing the opportunity to write my thesis at his chair for Software Engineering for Business Information Systems (SEBIS) and who helped to shape the topic with his valuable feedback.

Last but not least, I would like to thank my family, including my better half Hannah, and all my friends who have always supported me during this exciting journey. Thank you for your unconditional support and that I can always count on you.

Having the pleasure to work with passionate and committed people is invaluable. I am very grateful that I had the opportunity to do so as part of this master's thesis.

Abstract

Nowadays, large IT organizations are struggling to cope with unpredictable competitive environments due to rapidly changing customer needs, regulatory changes, and technological advancements. Thus, the ability of large IT organizations to react quickly to changes can be a significant competitive advantage. To achieve a high level of organization-wide agility, standardized processes and commitment from all stakeholders are necessary. Traditionally, top-down IT governance control mechanisms have been used to enforce a certain common direction within IT processes, thereby also aiming to achieve consistency and quality as well as to ensure compliance with legal requirements. However, these top-down control mechanisms do not fit well into increasingly widespread agile and lean environments. Scaling agile frameworks, such as Scaled Agile Framework or Disciplined Agile Delivery, recommend to use more lightweight and collaborative IT governance approaches. Yet, these frameworks do not provide enough concrete guidance to implement such a form of IT governance on a large scale.

We fill this gap by providing a collaborative approach to establish architecture principles and guidelines and a prototypical web application to enable and support the approach. The approach mainly revolves around a close collaboration between enterprise architects and agile teams, handling the full life cycle of architecture principles and guidelines together. The goal is to combine top-down (authoritarian) and bottom-up (self-governance) perspectives in order to leverage the advantages as well as mitigate the disadvantages of both sides.

To accomplish this, the thesis first analyzes the current state of existing research on how governance and architecture fit into modern, large-scale agile development environments. Afterwards, it first presents the results of a case study of a large global insurance company and then introduces the collaborative approach and tool support that aim to solve challenges identified in both research and practice. Subsequently, we evaluate the two solution artifacts of this thesis in expert interviews with fifteen participants from the case study organization. Finally, we summarize the key findings and give an outlook on possible further research. The results indicate a high approval for the approach and the tool support as well as a strong need for closer collaboration between both stakeholder groups, enterprise architects and agile teams.

Contents

Acknowledgments	iii
Abstract	iv
1. Introduction	1
1.1. Motivation	1
1.2. Research objectives	4
1.3. Research approach	5
2. Foundations	8
2.1. IT governance	8
2.1.1. Traditional IT governance	8
2.1.2. Modern governance approaches in the context of lean and agile	9
2.2. Enterprise architecture management	12
2.2.1. The role of EA in IT governance	13
2.3. Principles and guidelines	13
2.3.1. Definition and delimitation	14
2.3.2. Importance in enterprise architecture management	16
2.3.3. Value of principles	17
2.3.4. Generic process	17
2.4. Agile and lean development	24
2.4.1. Agile software development	24
2.4.2. Large-scale agile software development	25
2.4.3. Lean software development	26
2.4.4. Agile vs. lean software development	26
2.5. The interplay between EAM and large-scale agile software development	27
2.6. Gamification and social design	29
3. Related work	30
4. Case study	35
4.1. Case study design	35
4.2. Case description	38

4.3. Use of architecture principles and guidelines	39
4.4. Architecture communities	42
4.5. Role of the enterprise architect	42
4.6. Value contribution of enterprise architects	46
4.7. Summarized challenges	47
5. Collaborative approach to establish architecture principles and guidelines	50
5.1. Agile governance through collaboration	50
5.2. Guideline establishment approach with relevant stakeholders	52
5.2.1. Involved stakeholders and their role in the approach	52
5.2.2. Community goals and responsibilities	55
5.2.3. Collaborative process steps	57
5.3. Addressed challenges and solution requirements	66
6. Implementation	75
6.1. Motivation for a web application	75
6.2. Technical requirements and technology selection	77
6.3. Main views and core features	78
6.3.1. Overview of the guidelines of a specific team	78
6.3.2. Detail screen of the guideline of a specific team	81
6.3.3. Team dashboard	82
6.3.4. Team creation and guideline mapping	83
6.3.5. Team and guideline statistics	84
6.4. Features and possible extensions based on social design principles	84
6.5. System architecture	90
6.6. Class diagram	92
6.7. Possible extensions and next steps	94
7. Evaluation	97
7.1. Goal and methodology	97
7.2. Evaluation of the collaborative approach	99
7.2.1. Value for agile teams and enterprise architects	99
7.2.2. Challenges	101
7.2.3. Community activities and process steps	103
7.3. Evaluation of the prototype	114
7.3.1. Assessment of the main goals of core features	114
7.3.2. Usability assessment based on the System Usability Scale	116
7.4. Summary of the evaluation results	118

Contents

8. Discussion	120
8.1. Key findings	120
8.2. Limitations	122
9. Conclusion and future work	124
9.1. Summary	124
9.2. Future work	125
A. Appendix	127
A.1. Evaluation interviews	127
A.2. Semi-structured case study interviews	129
A.2.1. General information	129
A.2.2. Architecture principles	131
A.2.3. Architecture boards	132
A.2.4. Role of the enterprise architect	132
A.2.5. Value contribution of enterprise architects	134
Bibliography	137

1. Introduction

This chapter explains why adapting the use of architecture principles and guidelines to achieve agile and lean governance of large-scale agile development teams is a relevant and valuable research field to explore. Furthermore, it presents the main goal and research questions that this thesis addresses in Section 1.2 and demonstrates the applied methodology used to answer those research questions in Section 1.3.

1.1. Motivation

To understand why agile and lean governance of agile teams is a pressing issue, the following chapter looks at how large-scale agile is steadily growing in importance, but also how raising agile methods to a cross-team level comes with certain challenges. Subsequently, it justifies the proposal that architecture and governance, and specifically architecture principles and guidelines, can be a possible solution to those challenges, if adapted to agile settings.

Large-scale agile on the rise

Agile software development methods have been flourishing since the coining of the term "agile" with the creation of the "Agile Manifesto" in the year 2001. By now, they have reached a state of omnipresence. In the latest "State of agile survey" from 2018 [119], more than 97% of interviewed participants state that they use agile methods in their organization. Despite its prevalence, using agile methods is not yet a trivial, straightforward undertaking: Only 52% of interviewees have more than half of their teams practicing agile methods and values. Honda et al. [48] also highlight the fact that according to the "State of Agile" study results, "an overwhelming 84% of organizations are 'still maturing' in their agile practices", which stresses the ongoing need for research in the area of agile adoption and practical application. Since agile methods are increasingly moving out of their originally intended environment (small, co-located teams) into large-scale on the business and enterprise level [48], the question of how to improve the large-scale applicability of agile methods is becoming increasingly important. However, this development is not new. Already in 2010, "agile and large

projects” has been voted the number one “burning research question” by software practitioners [36], which shows the continuous importance and relevance of the topic.

Problems of agile on a large-scale

Whereas some studies show a positive impact of agile methods, e. g. when comparing them with their traditional counterparts in terms of effectiveness and successfulness [4], there are a lot of researchers questioning the viability of agile methods in large-scale endeavors. More specifically, in large-scale agile development, common agile methods are insufficient when developing complex systems [116]. They are especially problematic for large-scale system architectures and for systems incorporating existent and possibly evolving software architectures [10] or safety critical systems [115]. One of the common pitfalls is the lack of architecture and unclear role of architects [9], since according to Leffingwell and his colleagues, “some amount of architectural planning and governance is necessary to reliably produce and maintain such systems. Individuals teams, products and programs may not even have the visibility necessary to see how the larger, enterprise system needs to evolve.” [67] Other publications put emphasis on the lack of suitable IT-Governance methods that impact the ability to scale agile software development techniques, e. g. because of the issue of regulatory compliance [7] and concomitant audits [56]. Furthermore, high organizational agility is mainly found in companies with highly standardized processes and platforms [100]. Therefore, if an organization aims to achieve organization-wide agility, commitment from all involved stakeholders is needed, which in turn cannot be achieved without governance [50, 56].

Agile vs. (enterprise) architecture and governance

Responding to those challenges, the agile community started researching if and how methods from enterprise architecture, software architecture and IT-governance fit into agile methods. Whereas the “how” question is complex, without one, clear answer and still part of ongoing research, there is a multitude of researchers that strongly affirm the question if architecture and agile can and should be combined [10, 85, 101, 1]. The question of agile vs. architecture also goes hand in hand with the management and governance of agile teams and yet another appurtenant trade-off regarding governance: the self-organization vs. control of agile teams. Agile teams usually would like to be self-responsible and therefore have a lower acceptance for architectural guidance than traditional teams. This could be related to their concern that architects predetermine architectural and technical constraints, thereby limiting their freedom and slowing down their development speed. These challenges have to be taken into account when attempting to merge agile and architectural methods.

Collaborative use of architecture principles and guidelines to contribute to agile governance

The goal of this thesis is to help answer the question of how architecture and governance fit with agile development methods in large-scale endeavors. This thesis proposes the use of principles and guidelines, a key concept in enterprise architecture management (EAM), embedded in a collaborative approach that fits well to agile and lean values and principles. The incorporation of architecture principles is chosen because of their widespread use and high value. Specifically, they are deemed as highly valuable because according to Greefhorst and Proper [37], they

- "Fill the gap between high-level strategic intentions and concrete design decisions."
- "Document fundamental choices in an accessible form and ease communication."
- "Prevent analysis paralysis by focusing on the essence."

Albeit Greefhorst and Proper make a very valuable and comprehensive contribution to the field of architecture principles, they state that their focus is on a traditional top-down approach and stress the necessity for more research for more bottom-up driven processes and a higher collaboration in the creation and management of architecture principles [37]. This research gap is exactly where this thesis positions itself. We propose a collaborative approach for establishing architecture principles and guidelines and a software implementation that supports the collaborative approach. These two solution artifacts of this thesis are based on the findings of a case study we conducted at a large international insurance enterprise as well as findings from current research and related work. The proposed web application enables teams to join in on the guideline creation and management process, facilitates higher contribution and better feedback cycles, as well as starting points for a higher degree of automation. It also incorporates gamification and social design principles to motivate and encourage participating stakeholders to actively contribute.

As demonstrated in our analysis, a top-down governance approach in modern agile and lean environments is associated with many challenges. Therefore, our research makes a valuable contribution to the research gap on how to apply architecture principles in a lean and agile environment through a more bottom-up driven process. This also fits well to the research of Brosius et al. [19], who state that the main proportion of enterprise architecture research promotes to enforce and control compliance with enterprise architecture. But for further research, there is currently a high need for more research that is less concerned with enforcing or controlling, but rather focuses on empowering and supporting stakeholders to achieve the intended outcome of enterprise architecture. They further conclude that "this may be realized, for example, by granting more autonomy as well as more decision-making authority to local stakeholders" [19].

1.2. Research objectives

To incorporate principles and guidelines in large-scale agile development environments and adapt them to facilitate a higher degree of collaboration and acceptance between different stakeholders, we divide the overall research goal into the following three research questions:

Research question 1: How does governance and enterprise architecture fit in modern large-scale agile and lean development environments?

To answer the first research question, existing literature on IT-governance, enterprise architecture, large-scale agile development and the interplay between these areas are analyzed. In addition, there is a detailed look at architecture principles and guidelines as an established artifact from enterprise architecture and the existing processes for using architecture principles and guidelines. Furthermore, a case study is used to enrich the results from current literature, with a focus on identifying the challenges that occur in the area of governance in large-scale agile software development in actual daily practice.

Research question 2: How can agile teams and enterprise architects collaboratively establish and manage architecture principles and guidelines in large-scale agile software development?

Since existing approaches related to architecture principles and guidelines primarily focus on top-down driven perspectives and processes [37], the second research question mainly revolves around the issue of how a combination of top-down and bottom-up processes for establishing architecture principles and guidelines could look like. This is closely linked to the question of control versus autonomy of agile teams and how agile teams can become the main stakeholder in establishing architecture principles and guidelines, therefore resulting in a form of self-governance, without losing the support and oversight from enterprise architects. To answer this research question, we build on the existing practical approach by Greefhorst and Proper [37, 38] and extend it by proposing a collaborative approach, taking into account the findings from the literature, related work and practical insights from the case study. Finally, the approach is evaluated through expert interviews with fifteen participants of the case organization.

Research question 3: How can the collaborative approach for establishing architecture principles and guidelines be supported and further enhanced by a software implementation?

To answer this question, we develop a software implementation to support the collaborative approach and further address the identified challenges. We focus on providing a software implementation to support the collaborative approach because of specific reasons and requirements outlined in Section 6.1. It sets the basis for more automation in the future and incorporates social design principles and gamification elements to encourage contribution. The tool support is part of the evaluation with fifteen experts as well, mainly focusing on comparing the current status with the resulting situation using the web application to assess the achieved improvement, as well as an evaluation of the usability of the tool support.

1.3. Research approach

The following section gives an overview of the overall research methodology applied in this thesis.

This master's thesis relies on two main research approaches: the design science paradigm and the case study approach. The design science paradigm, which was introduced and popularized in the field of information systems research by Hevner et al. [46] and further refined and completed with the suggestion of a process to carry out design science research by Peffers et al. [89], is the main approach of this thesis. It centers around creating and evaluating new and innovative artifacts within a problem domain, intending to solve identified organizational problems. Thereby, applied specifically to our thesis, it sets the scene for systematically developing a tool-supported, collaborative approach for agile and lean governance in large-scale endeavors with the use of principles and guidelines. The two resulting artifacts are:

- (i) the concept for a collaborative approach for establishing architecture principles and guidelines
- (ii) the prototypical implementation of the tool-support for the approach, a web-based application

Furthermore, this thesis incorporates a case study with our partner company within the design science approach. Instead of using the case study primarily as a method for design evaluation of the created artifact, as originally proposed by Hevner et al. [46], we include the case study results throughout the whole design science approach. This is inspired by modern, human-centered design thinking approaches which revolve around continuously involving target users from the beginning and collecting requirements and feedback before and during the actual artifact development, not only after the design and development phase. This approach fits well to the "action design research" approach proposed by Sein et al. [106]. In contrast to the regular design science approaches which

propose rather separated and sequenced key steps, specifically an "evaluation" phase after the "design and development" [89], action design research acknowledges the need for interweavingly building the artifact and continuously evaluating it. It emphasizes that the resulting artifacts will not only mirror the design intentions of the researchers, but is repeatedly shaped by organizational use, perspectives and participants and by the outcomes of concurrent evaluation [106]. In regards to the classification for case studies proposed by Yin [130], the present case study is a single-case study. More details on our case design can be found in Section 4.1. The case study approach itself is based on the insights of Runeson and Höst, who provide a case study methodology and guidelines for researchers in the area of software engineering [102].

Figure 1.1 shows an overview of our research approach, based on Hevner [46], Peffers [89], Sein [106] and Runeson and Höst [102], adapted to our research. In addition to the methodological procedure, Figure 1.1 also shows the mapping between the design science phases and the chapters of this thesis.

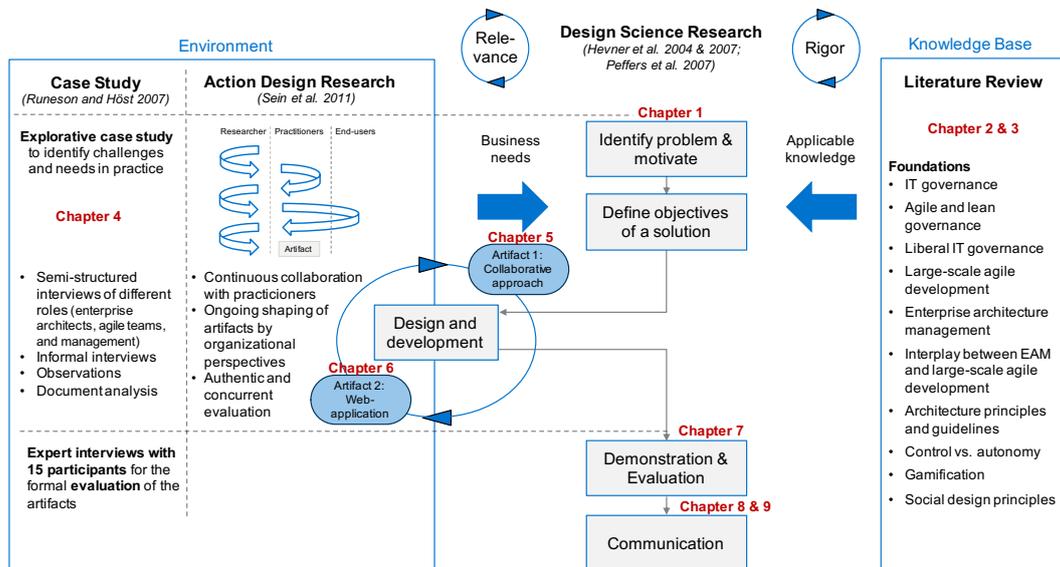


Figure 1.1.: Research approach overview and mapping with thesis chapters

By combining the research approaches, we aim to find the right balance between ensuring practical relevance and applicability as well as achieving generalizability of results and importance for research. The case study and action design research provide the relevance, whereas the literature review and related work contribute the applicable knowledge for the necessary rigor of our research. In regards to the characteristics of different research methodologies described by Runeson and Höst, our research is of exploratory and improving nature [102]. Therefore, our primary data is of qualitative

nature and the research design is rather flexible than fixed [102].

The structure of the thesis is as follows. Chapter 1 introduces and motivates the thesis topic and defines the research objectives and methodology. Chapter 2 describes and summarizes the findings of existing literature. In addition, Chapter 3 presents additional relevant related work. Chapter 4 details the case study conducted during this thesis. Chapters 5 and 6 present the two solution artifacts of this thesis, the collaborative approach and the tool-support implementation. Chapter 7 demonstrates the main results of the evaluation. Chapters 8 and 9 summarize key findings, limitations and unveil an outlook for further research and next steps.

2. Foundations

This chapter provides the theoretical foundations for the remaining chapters of this thesis. The goal is to present existing research and findings, establish a common understanding of the relevant terms and concepts and thereby lay the groundwork for building on top of these subjects during the subsequent parts of the thesis. The main focus is on exploring IT governance (Section 2.1), enterprise architecture (Section 2.2), architecture principles and guidelines (Section 2.3), (large-scale) agile and lean development (Section 2.4), and pointing out the relationships between those areas (Section 2.5). Finally, in Section 2.6, it gives a brief outline of gamification and social design aspects that are used later in the thesis.

2.1. IT governance

IT governance is a subset of corporate governance, which is focusing on leadership and control of the organization to achieve responsible and long-term value creation [129, 80]. Corporate governance is an important part of business management [80, 128]. The first of the following two subsections defines traditional IT governance. The second subsection dives deeper into IT governance within an agile and lean context.

2.1.1. Traditional IT governance

The specific term "IT governance" originated in the late 1990s when Brown [20] and Sambamurthy and Zmud [103] first started using the term in reference to an IT governance framework [66]. The ISO/IEC standard 38500:2015 defines the governance of IT as "a subset or domain of organizational governance, or in the case of a corporation, corporate governance" [53].

In line with the ISO standard, Winter et al. state that IT governance takes the role of corporate governance specifically to the IT area [128]. One of the most cited definitions of IT governance besides the ISO-standard is the definition by Weill, who defines the goal of IT Governance as "specifying the decision rights and accountability framework to encourage desirable behavior in the use of IT" [124]. To achieve the encouragement of "desirable behavior in the use of IT", IT governance includes policies and methods

to ensure that IT is aligned with business goals, that IT resources are properly and responsibly used as well as the management and monitoring of risks [80].

After the coining of the term, the importance of IT governance was accelerated by the enactment of the "Sarbanes-Oxley Act" in 2002, urging companies to put much more effort into compliance [68]. But IT-governance has also become crucial due to the increasing and overarching use of technology, which results in high IT investments [55] and a critical dependency on IT [118, 90]. Therefore, companies can no longer afford bad IT governance [123, 90]. IT governance also plays a key role in achieving IT-based synergies, which can lead to reduced expenditures, higher knowledge sharing and may also achieve higher innovative capabilities [61, 110].

The practical adaption of IT governance is well established, especially through COBIT, which is one of the most widely used and adapted reference models in the area of IT governance [128, 80, 90, 95, 61]. COBIT 5 defines governance as follows [52]: "Governance ensures that enterprise objectives are achieved by evaluating stakeholder needs, conditions and options, setting direction through prioritization and decisions making; and monitoring performance, compliance and progress against agreed-on direction and objectives." Thanks to the widespread use of IT governance, researchers were also able to analyze the impact and value of IT governance. Multiple studies indicate a significantly (up to 40%) increased return on IT investments [126, 69]. Lazic et al., who researched the business impact of IT governance and specifically the reasons behind the impact, state that the positive impact results from increasing both the relatedness of IT and the relatedness of business processes, which also improves the inter-relatedness between IT and business processes [65].

2.1.2. Modern governance approaches in the context of lean and agile

According to Kude and his colleagues, in an organization that is striving for cost leadership and efficiency, IT governance should aim towards control and regulation to achieve its' goal in a rather short time [61]. Nevertheless, they stress that in times of increasing influence of IT, consensus-oriented IT governance capabilities are important or even required, especially for fostering innovation in IT and recombining core competencies to create value [61]. Furthermore, the traditional, top-down, authoritarian control approach to governance does not seem to fit well to agile and lean values and principles, which put more emphasis on self-organization and self-responsibility. The key challenges when applying traditional governance in agile environments are mainly the lack of an agile approach [71, 72, 94] as well as the lack of appreciation of people in governance processes [71]. Because agile and lean development approaches account for a large part of today's software development efforts [122, 48], a better suitable, modern approach to IT governance is desirable. Ambler and Kroll from IBM state in

their whitepaper titled "Lean development governance" that effective governance is not about command and control, but enabling the desired behavior through collaborative and supportive methods [8]. Furthermore, they conclude that it is far more productive to motivate people to do the right thing instead of forcing them to do so [8]. Two terms that describe this kind of thinking are "Lean Governance" and "Agile Governance". "Lean Governance" appears as a term that is used mainly in practice, indicated by multiple white-papers on the topic [8, 73], as well as the use of the term in popular scaling agile frameworks, namely the "Disciplined Agile (DA 2.0)" [5] framework and the "Scaling Agile Framework (SAFe)" [104], but barely any research publications. "Agile Governance" on the other hand is also addressed by researchers [72, 23, 94, 56]. Some scaling agile frameworks, namely SAFe, EADAGP and DA 2.0, acknowledge the need for not slowing down teams, but still take in account necessary constraints. Therefore, they recommend modernizing governance models, resulting in more light-weight, collaborative and decentralized approaches [117]. Nevertheless, they do not provide enough guidance how such approaches could be applied in practice. Furthermore, most decisions are still made top-down without using input and qualitative feedback from teams [50].

Often, the two terms "Lean Governance and Agile Governance" are combined and used interchangeably. Luna et al. state that they base their work on agile governance on the definition by Kruchten [60], which describes agility as "the ability of an organization to react to changes in its environment faster than the rate of these changes" [60], so that they can unify agile and lean approaches by simply using the term agile and not differentiating between agile and lean governance [56]. Based on their analysis, they define agile governance as "the ability of human societies to sense, adapt and respond rapidly and sustainably to changes in its environment, by means of the coordinated combination of agile and lean capabilities with governance capabilities, in order to deliver value faster, better, and cheaper to their core business." [56]

Because of the similarity of both the concepts of lean and agile governance and the lack of any clear differentiation in research and practice, we will also treat the terms "agile governance" and "lean governance" as synonyms.

Luna et al. acknowledge that applying agility and governance together might seem counter-intuitive at first, but they stress that to achieve the goal of business agility, there has to be commitment from all parts of an organization, "which in turn cannot be achieved without governance" [56]. This claim fits well to the research of Ross, Weill and Robertson, who describe that organizational flexibility and agility requires limiting local flexibility and enforcing standardization of parts of the organization at first. This then later at a higher maturity level enables a much higher organizational flexibility and in the end also achieves higher local flexibility again [100], as illustrated in Figure 2.1.

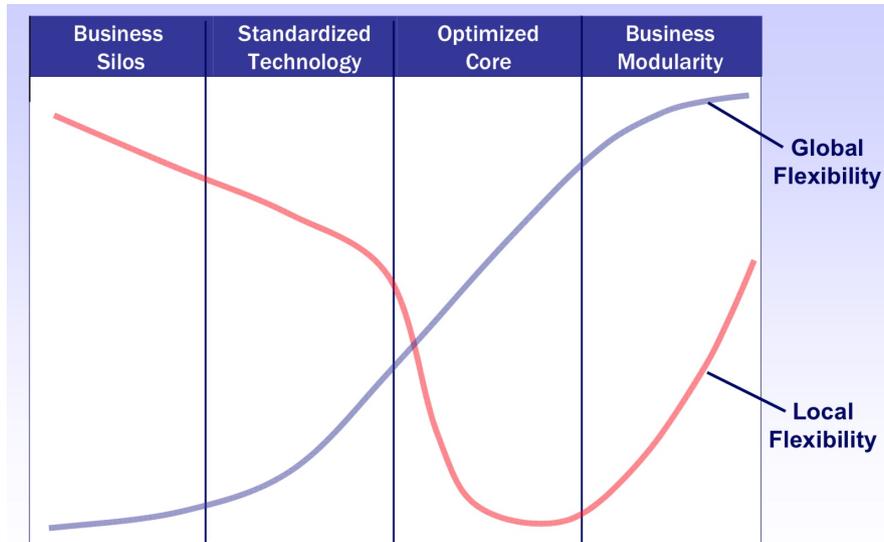


Figure 2.1.: Changes in organizational flexibility through the architecture stages by Ross, Weill and Robertson [100]

Rob and Weill summarize their findings in what they call the "Agility paradox". It describes that firms with high organizational agility more often have a higher degree of standardized and digitized business processes and platforms [100]. Therefore, governance is not in the way of reaching organizational agility, but an enabler to do so, as long as the traditional, controlling top-down approach is reconsidered and transformed into a more collaborative and supportive approach, which is more compatible with lean and agile principles. The goal to achieve a more collaborative and supportive approach to governance, instead of a strict control approach, can also be described as "liberal governance", compared to its' traditional counterpart, the "sovereign governance" [66]. In their analysis, Leclercq-Vandelannoitte and her colleagues stress that the changing technological and social environment that surround the organizations and "[...] the autonomy of individual users and the changing nature of IT" disrupted the classic, sovereign governance approach [66]. The vision of a liberal model of IT governance is that people are active in their own self-government [66]. Hence, not only top-down laws are required, but methods that actually contribute to self-governance. This fits very well to the research of Luna et al. who infer that people are a central element of governance and who see "people's engagement" as one of the key values in their proposed "Agile Governance Manifesto" [71].

This all leads to the question of how much decision-making ability teams should have and the crucial trade-off between control and autonomy of agile teams. This trade-off is

also recognized by current research: Uludağ et al. point out the need to find the right balance between centralized and decentralized architectural decision-making [117]. They explain that decision making higher up in the chain of authority causes delays and can decrease the usefulness of the architectural decisions [117]. Horlach et al. conclude in their research that further investigation is necessary on how to balance autonomy and authority when integrating agility and governance [50].

In regards of top-down and bottom-up governance processes, we speak of "top-down" describing governance efforts that are mainly driven by entities or roles higher in the organizational hierarchy, setting rules, controlling and communicating to groups lower in the organizational hierarchy. This is what can be described as "traditional" or "sovereign governance" [66]. With "bottom-up" on the other hand, we describe governance efforts mainly driven by those entities directly affected by the governance, or what we therefore call "self-governance", in line with other researchers. Horlach et al. use the terms similar to our understanding and summarize the two approaches as "authority-led" (top-down) and "autonomy-led" (bottom up) [50].

2.2. Enterprise architecture management

The ISO, IEC and IEEE describe architecture in their current Standard 42010 as the "fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" [54]. Derived therefrom, enterprise architecture can be seen as the fundamental structuring of an enterprise [3, 76, 127]. While the terms "enterprise architecture" and the term "architecture" are mixed and used as synonyms by some authors, the main difference lies in the explicit incorporation of more business-related artifacts within an enterprise architecture. Enterprise architecture thereby extends traditional IT architecture with aspects such as organizational goals, products and services or markets and competitors [127]. This holistic approach leads to better business-IT alignment [128, 59, 125]. Another important aspect is that enterprise architecture (management) deals with the current, planned and target state in the future and the transitions in-between, leading from the current state to the target state [112, 3, 43, 21]. This aspect is also reflected in the enterprise architecture definition of Greefhorst and Proper, who state that "Enterprise architecture is positioned as an instrument to articulate an enterprise's future direction, while serving as a coordination and steering mechanism toward the actual transformation of the enterprise" [37]. A predominant number of enterprise architecture frameworks differentiate between multiple architecture layers and views because of the broad scope and the resulting high number of artifacts [127]. Examples for frameworks with a large usage in the past are TOGAF and the ARIS framework, with

layers like the business strategy layer, organization/business process layer, integration layer, software/data layer and IT infrastructure layer [128].

In their state-of-the-art analysis report on enterprise architecture management literature, Buckl and Schweda use the terms "enterprise architecture" and "enterprise architecture management" as synonyms, but emphasize that the term "enterprise architecture" was coined in the year 1993 by both Henderson and Venkatraman [44], as well as Spewak and Hill [39], and later transitioned into the more recent term "enterprise architecture management" [21].

In recent years, agile and lean approaches also increased in importance within the field of enterprise architecture, resulting in calls for a more agile, adaptive approach on enterprise architecture [58]. One view is that "EA should be an embedded organizational capacity and shared concern embraced by everyone, not just some master planners and designers in an EA ivory tower." [58]

Key concepts in the field of enterprise architecture include stakeholders and their concerns, principles, models, views and frameworks [37, 63]. The concept of principles is the most relevant element of enterprise architecture management for the scope of this thesis and will be analyzed in more detail later on in Section 2.3.

2.2.1. The role of EA in IT governance

Even if there is certainly a large overlap between the activities and relevant artifacts of IT governance and enterprise architecture, the main difference is in their focus areas: IT governance controls, enterprise architecture informs and is a basis for decision making and control. Thus, enterprise architecture can be seen as a basis for IT governance [83]. Winter and Schelp are in line with this view: They see enterprise architecture as an important source of information regarding compliance [128]. In recent years, an increasing number of researchers have dealt with the question of how enterprise architecture governance should look like, further showing the relatedness of the disciplines. Korhonen et al. [58] describe one of the possible interpretations of EA as the following: "Enterprise Architecture is a process meant to identify (and govern) the desirable changes to the IT resource landscape in order to achieve coherence (alignment) between desirable business objectives or outcomes as well as to enable new business objectives and foster new business opportunities." [58]

2.3. Principles and guidelines

The next sections detail the results of our literature review on architecture principles and guidelines. The section will deal with both the terms "principles" and "guidelines", because as the analysis shows, they are interwoven concepts, which are often used

interchangeably, but there are also existing efforts for delimiting the two terms. We start of by illustrating the relationship of principles and enterprise architecture and then move on to define principles and guidelines for the use in the rest of the thesis. Once we have clarified what principles in the enterprise architecture context are, and what they are not, we dive deeper in the process of creating and managing principles. The structure of this process is based on the practical approach to architecture principles proposed by Greefhorst and Proper [38]. Their generic process includes determining drivers, determining principles, specifying, classifying, validating and accepting those principles as well as applying them, manage their compliance and handle changes [38].

2.3.1. Definition and delimitation

Principles in general are defined in the dictionary as a rule or code of conduct or a comprehensive and fundamental law, doctrine, or assumption [79]. Guidelines are defined as an indication or outline of policy or conduct [78]. Guidelines have been used in the IT environment in several domains and they can be based on practical experience or derived from research [90]. To focus on the most relevant definitions and research for our thesis, we will focus on principles and guidelines in the context of enterprise architecture management, where principles and guidelines are mostly summarized under the term "architecture principles". To properly contextualize architecture principles, it is important to make the distinction between scientific and normative principles. According to Greefhorst and Proper, scientific principles are "a law or fact of nature underlying the working of an artifact" [38]. Normative principles however are "not enforced by nature. but require explicit attention to be enforced." Architecture principles belong to the class of normative principles [38].

There is no single, widely applicable definition of enterprise architecture principles, however, the general intend of principles is to set a constraint on the design space of enterprises and (information) systems and guide design decisions [87, 108, 40, 16]. According to ANSI/IEEE STD 1471-2000, principles are used for the governance of architecture design and evolution over the life cycle of a system [54].

To clearly delineate enterprise architecture principles, it is worth noting how the term is often misunderstood in our context. For this purpose, we adduce the analysis of Haki and Legner [40] regarding three common confusions of enterprise architecture principles. They state that principles are not enterprise architecture management goals, neither enterprise architecture management practices nor more low-level governance means as guidelines and standards. The difference is that enterprise architecture management goals do not directly limit the design space or guide design decisions (they do however have a strong impact on the rationales of principles). Enterprise architecture management practices on the other hand describe the organizational

process to conduct enterprise architecture management, with best practices and success factors. Yet again, they do not contribute to limiting design space or guiding design decisions. Regarding the third point, the confusion of principles with guidelines and standards, Haki and Legner state that they see principles as having a higher level granularity, concerning high-level strategic decisions, whereas standards and guidelines represent more low-level governance means [40]. Nevertheless, the terms "principles" and "guidelines" are used interchangeably and thereby implicitly seen as synonyms in some publications and frameworks: e. g. TOGAF defines principles as "general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an enterprise sets about fulfilling its mission" [112]. Also, Richardson et al., who were the first researchers to investigate enterprise architecture principles according to an analysis by Haki and Legner [40], define principles as guidelines and rationales for continuously examining the IT target plan [97]. Roos and Mentz define guidelines as "a general rule, principle, or piece of advice" [99]. These examples illustrate how the definitions of the two terms often include each other, further showing how interwoven the two concepts are. Interestingly, Greefhorst and Proper do not clearly differentiate between principles and guidelines in their book on architecture principles either, but they do so in an earlier publication, where they make the distinction that "[...] guidelines are more specific than architecture principles" [93]. Perhaps, this lack of further distinction has to do with their broad interpretation of principles, which, in their view, can have different levels of specificity [37], thereby already including "guidelines" and superseding the additional term. This would also fit to the view of Op't Land et al. who see "guidelines" as a possible role that principles can take. They state that guidelines are "properties of (classes of) a system that are specific enough to provide guidance to operational behavior [...]" [63]. The importance of this specificity is also recognized by Greefhorst & Proper who state that when enterprises would like to use principles to actually limit design freedom, they need to be specific enough and formulated in a way that allows assessing the compliance with those principles [37].

Because the focus of this thesis is not only the strategic aspect of principles, but above all on the practical application and the impact on actual implementation, we would like to forego merging the two terms, but instead explicitly differentiate the two terms: To highlight the importance of the right level of specificity of principles, we will mainly use the term *guideline* when aiming to emphasize that we are referring to an instance of a principle that is specific enough to apply guidance for actual implementation. The term *principle* is used to also include more generic principles that are not necessarily suitable to directly guide implementation, but that are more strategic and can be used to derive one or multiple more specific guidelines from. This interpretation is also in line with the three common misconceptions of EA principles analyzed by Haki and

Legner [40], since the confusion of principles with guidelines is one of those three common misconceptions, as explained earlier. Other related terms to principles and guidelines are rules, policies, standards, norms, credos and regulations. To achieve consistency, we will stick to the terms principles and guidelines during this thesis.

2.3.2. Importance in enterprise architecture management

This section illustrates the importance of principles and guidelines in enterprise architecture management. Weill and Ross name "principles" as one of five important type of IT decisions in IT governance and stress that the joint input of business and IT should be used to combine strategic business decisions with more technical and organizational aspects [126]. According to an expert study conducted by Haki and Legner, most of their questioned experts regard principles as very useful. They conclude that "[...] most experts believe that EA principles should be an integral part and essential element of EA and should be considered a necessity" [40]. The most important reason to utilize enterprise architecture principles is as a part of enterprise architecture governance for achieving coherence and harmonization [40]. Lumor et al. state that "the importance of EA principles, in the development of flexible architecture and coherent systems, has long been identified" as well as "the importance of EA principles is clear" [70]. Some researchers even position principles as the essence of architecture [37, 49]. Furthermore, multiple authors mention the influence of principles and guidelines in enterprise architecture decision-making [99, 84, 109, 92].

The following listing shows various, exemplary definitions of architecture and enterprise architecture management and highlights the recurring occurrence of principles and guidelines in those definitions, further demonstrating the high importance of those concepts within enterprise architecture and IT architecture:

- "EAM is a management practice that establishes, maintains and uses a coherent set of **guidelines, architecture principles** and governance styles to achieve enterprise's vision and strategy" [57] based on [2]
- "An architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the **principles** guiding its design and evolution" [54].
- "Architecture has two meanings depending upon its contextual usage: (1) A formal description of a system, or a detailed plan of the system at component level to guide its implementation; (2) The structure of components, their interrelationships, and the **principles and guidelines** governing their design and evolution over time." [112]

2.3.3. Value of principles

Now that we have shown that the importance of principles is underpinned by multiple researchers and practitioners, we will summarize of what exactly makes principles and guidelines important and what value their usage can achieve. Greefhorst and Proper [37] explore the value of principles and summarize the following benefits of architecture principles. They state that architecture principles bridge the gap between strategic goals and actual design decisions affecting implementation. Principles ensure that EA is future oriented and deals with the target state, but at the same time prevent "analysis paralysis" by focusing on the most significant, core aspects. This focus on the essential requirements enables enterprises to carefully decide on what they would like to design and govern in a top-down approach and what they would like to leave up to emergence. Another benefit is that they "document fundamental choices in an accessible form, and ease communication with all those affected" [37]. Furthermore, principles contribute to a form of continuity and stability, in an environment of change and uncertainty. Principles do not only exist for solely providing constraints. In fact, they contain important design knowledge and act as a source of inspiration for discussion and the creation of other artifacts [37].

With regards to governance, Greefhorst and Proper see architecture principles as "the primary enablers for an effective architecture governance" [38]. Compared to architecture models, Greefhorst and Proper assess principles as the more suited governance instrument because they give a better sense of what is important, less room for interpretation and give more insights for implementation and compliance than models. Additionally, they provide better opportunity for iterative, small-scale releases due to their self-contained character and higher independence from each other [38].

2.3.4. Generic process

This section summarizes a generic approach and process for the development and use of architecture guidelines, as proposed by Greefhorst and Proper. The process by Greefhorst and Proper is chosen because of their extensive and renowned research, providing a more detailed and practical approach compared to other sources. Furthermore, they have developed and validated their approach over the course of multiple years [38]. Their generic process includes determining drivers, determining principles, specifying, classifying, validating and accepting those principles as well as applying them, manage their compliance and handle changes [37], as shown in Figure 2.2. Since they mainly use the term "principles", we will stick to their wording in the description of the process. Later in the thesis, we will incorporate the use of guidelines into the process. In accordance to the previously described terminology, the term guideline

hints to a more specific version of a principle.

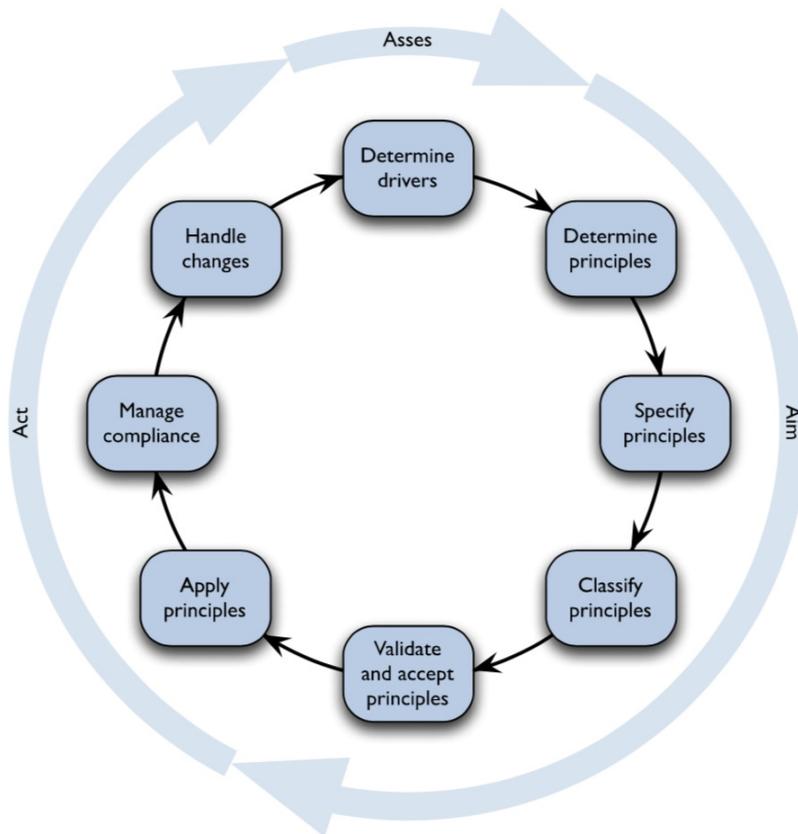


Figure 2.2.: Generic process for handling architecture principles by Greefhorst and Proper [37]

Determining drivers

Determining drivers revolves around analyzing relevant sources and collecting suitable input for deriving architecture principles, e. g. goals and objectives, values, issues, risks, potential rewards and constraints. Table 2.1 shows a short description and an example for each of the drivers, for illustration purposes.

It is the responsibility of the architects to identify those drivers, keep them up to date and ensure their quality. In terms of quality, the drivers especially require a high clarity, thereby preventing ambiguity. Drivers are not always explicitly documented and have to be gathered from multiple stakeholders. Since drivers are often "poorly

2. Foundations

Name	Description	Example	Question to derive principle from driver
Goals & Objectives	Targets that stakeholders within and outside an enterprise seek to meet. Many of these will be embedded in the strategy of the enterprise.	"Decrease costs" (high-level) or "Decrease IT development costs with 10% within one year." (more specific)	What is needed to attain the goal or objective?
Values	Fundamental beliefs shared between people in an enterprise. Values are expressed in terms of quality attributes.	Reliability, trustworthiness, transparency, sustainability, efficiency, flexibility, privacy (see ISO 9126 and IEEE 1061 for more)	What is needed to realize this value?
Issues	Issues are anything that hinders an enterprise in reaching its goals and problems that the organization face. They exist at all levels, from strategic to tactical and operational.	An example of an operational issue is "IT systems do not reach the availability requirements as set forth in the Service Level Agreement"	What is needed to solve the issue?
Risks	Problems that may occur in the future and that hinder the enterprise in reaching its goals. They are essentially issues that may occur in the future.	There are single points of failure in the infrastructure that may lead to unavailability of IT systems.	What is needed to minimize the probability or the impact of the risk?
Potential rewards	Business opportunities, chances and their potential reward for enterprises. In this sense, a potential reward is the inverse of a risk.	No specific example given	What is needed to attain the potential reward?
Constraints	Constraints are defined by others and cannot be changed by the architect. They may come from outside the enterprise, such as laws, policies and regulations provided by government, or they may also come from (senior) management.	All non-core activities will be outsourced.	What is needed to enforce the constraint?

Table 2.1.: Summary of driver description and questions to derive principles, as proposed by Greefhorst and Proper [37]

documented or the documentation is hard to find" [38], there might even be a need for conducting analysis, interviews, organizing workshops or organizing questionnaires to further refine those drivers. Greefhorst and Proper recommend focusing on goals and issues as the main drivers first, to avoid an overly complex process that takes in account all types of drivers, but is no longer applicable due to time and capacity constraints in practical environments [38].

Determining principles

The next step in the process revolves around the question of how to translate the drivers identified in the previous step into architecture principles. Greefhorst and Proper see three basic activities during the determination of principles that are used in combination: Generate candidate principles, select relevant principles and formulate principle statements [38].

Generate candidate principles

Generating a list of principle candidates in the first step can be achieved with the help of three approaches: Deriving principles from the identified drivers, elicitation of domain knowledge and harvesting existing principles [38]. Deriving principles from drivers makes sure that those principles are properly motivated. This is especially valuable in the communication with stakeholders and to receive their commitment to those principles. Refer to Table 2.1 to review the questions that can be used to derive principles from the different kinds of drivers. Domain knowledge is crucial for a deep understanding of drivers and to develop proper solutions. The gathering of in-depth knowledge from a domain expert can be necessary to further enrich the knowledge that architects contribute to the process. Harvesting existing architecture principles can contribute to creating a starting set for new architecture principles, but it should not be used alone, since some of the most important principles could be overlooked. The principles in question during this step are mainly the ones that have not been formally agreed on, but exist implicitly in some sources. One of the most interesting sources are solution architectures that may contain principles which can be reused. In the best case, a repository with material and information based on previous architecture experiences already exists. Otherwise, architects could try to set up such a repository based on previous projects [38].

Select relevant principles

The selection process starts based on the result from the previous phase: the list of architecture principle candidates. This list now needs to be filtered to ensure that only relevant principles are included. This selection can be seen as a way of prioritization.

Limiting the number of principles is important to reduce the time that is required from stakeholders and prevent over-complexity and limited accessibility. Greefhorst and Proper do not set a specific limit for the amount of principles, but stress that you should not be afraid to "throw away architecture principles that do not really express and essential choice and/or are not specific enough for the organizational context" [38].

A crucial activity during the selection is to filter out architecture principle candidates that are not actually architecture principles, but rather actions, requirements, strategic decisions, business principles, IT principles and more detailed design principles. Greefhorst and Proper propose the following questions to filter out other types of candidates that usually end up in a candidate list [38]:

- "Does it describe a functionality that is needed? In that case it is probably a (functional) requirement."
- "Does it describe something that needs to be done? If it does, then it is probably an action."
- "Are there objective arguments that support it? If it does not, then it is probably a (potential) strategic decision or business principle?"
- "Does it have impact on the design of the organization and/or the IT environment? If it does not, then it is probably a business principle (if it influences the daily business operations) or an IT principle (if it influences the daily IT operations)."
- "Does it have impact on the design of multiple systems? If it does not, then it is probably a more detailed design principle or design decision."

Formulate principle statements

In this step, the selected principles should be refined. Specifically, they should be analyzed regarding their specificity and should be generalized or specified as needed. The goal is for these principles to cover as many solutions as possible that match the scope of the architecture, while still guiding actual implementation. This is important, as too much generalization can be counter-productive because it undermines the credibility of the architecture and often results in a lack of relevance and applicability. This step can be conducted by single architects, but the end result should be validated within a group.

Specifying principles

In the previous step, the general principle statements or descriptions were formulated. In this step, the focus is on specifying those principles further by describing all attributes

that have been selected to describe an architecture principle. In an exemplary use case presented by Greefhorst and Proper, they state that "the drafting of architecture principle specifications was performed in the same workshop in which they were identified" [38]. This indicates that the "formulating principle statements" step from the previous sub-process could sometimes be partly merged with the principle specification. They also explain that full, detailed specifications might not always be necessary. Depending on architecture maturity and organizational culture, it may be sufficient in certain situations to forego specifying the principle and just use it to achieve a common understanding and commitment for certain issues, instead of actually restricting design freedom. In all other cases, it is advisable to start with the basic structure containing the rationale and implications and then iteratively add other attributes later. The selection of the relevant attributes depend on the specific requirements and needs in the organization. Lastly, Greefhorst and Proper also note the importance of what they call "guiding architecture principles", which is another kind of prioritization, indicating the most fundamental principles, often characterized by being the hardest to change and the closest to the identified drivers. They propose the rule of thumb of having no more than ten guiding architecture principles [38].

Classifying principles

It can be useful to classify the architecture principles after their specification. This is especially the case when there is a large number of architecture principles. Greefhorst and Proper differentiate the handling of architecture principles in terms of what they call a "low ambition level" to a "high ambition level" [38]. The low ambition level describes the handling of architecture principles when the number of principles is limited and the adherence to them is not formalized. The high ambition level on the other hand could contain "[...] hundreds of architecture principles, scattered around a large number of documents, owned by different stakeholders and governed by a formalized process" [38]. This level describes the situation where classifying principles is of high importance. The value of classifying principles is an increased accessibility and maintainability, by providing a navigation structure and overview as an entry point to the principle catalogue as well as a better categorization or clustering of principles.

Validating and accepting principles

Because of their high importance, a high quality of principles has to be ensured. Important to note here is that all the previous steps should include some sort of validation and at the same time can already be regarded as a validation on the way to high-quality principles. Nevertheless, Greefhorst and Proper explicitly point out

validation as a formal subprocess to stress its importance and the necessity of a quality gateway [38]. The validating process itself can be highly standardized and include specific roles, e. g. discussing and agreeing on the principles in an architectural board with management representatives of all major departments [38].

Apply principles

Greefhorst and Proper conclude that there is surprisingly little guidance on how stakeholders actually use architecture principles in the creation of their own artifacts [38]. One aspect is the need for validating solution requirements in regards to the compliance with architecture principles. Furthermore, architecture principles can also lead to new solution requirements. This conversion depends on the knowledge and experience of the involved stakeholders. In addition, Greefhorst and Proper stress that they distinguish two important types of transformations. One is the derivation, aiming to find more specific statements that realize an architecture principle. Here, it is important to document this transformation in the solution architecture, since this enables to trace the way how an architecture principle is actually implemented. This information can also be used in a compliance review. The second transformation is from principles to models and their visual representations, e. g. diagrams. In this case, architecture principles can be used to reason and distinguish different elements or types of elements in the models [38].

Manage compliance

It is important to manage and monitor compliance with the agreed architecture principles. There are often good reasons to deviate from the standards defined by architecture principles, as not all circumstances can be taken into account during the specification. Therefore, it is important to draw on findings from specific situations and use that insight to adapt the architecture principle specifications accordingly. It gives the management the opportunity to recognize potential problems and witness the actual impact of the architecture and governance. Greefhorst and Proper emphasize that an effective architecture compliance process must be conducted several times in the project life-cycle, spanning from the project initialization to the final completion, to harvest all the project insights. They also suggest that the architecture compliance process needs an overall architectural governance framework, which "[...] implies a clear architectural organization such as an architecture board, and clear roles and responsibilities" [38].

Handling change

Even if architecture principles are supposed to be relatively stable, there are always influencing factors that require the change or rework of certain principles. This can be experience from compliance reviews or other processes and sources. The responsibility of the architect is to monitor the presented potential drivers and take action accordingly, if necessary. In general, small changes can be applied by directly altering a principle, bigger changes on the other hand might require a new principle. The advantage of principles is that they are mostly self-contained, which provides the opportunity for small-scale, iterative releases. Lastly, Greefhorst and Proper also propose a feedback mechanism where people can comment on architecture principles, request changes or discuss with peers on specific experiences [38].

2.4. Agile and lean development

The next section dives into agile software development, large-scale agile software development and the relationship between lean and agile.

2.4.1. Agile software development

Agile software development originated as a counter-movement to the increasingly complex processes, project management, tools and documentation, which at the time (in the 70s and 80s) was necessary to provide the software discipline with more engineering rigor, but progressed more and more into losing the human side of software engineering [48]. The actual, wide-spreading coining of the term "agile" is mostly linked to the publishing of the Agile Manifesto [14] in 2001 [122]. The Agile manifesto clearly defines four core values and twelve agile principles [14], whereas the subsequent interpretation and application of "agile" is much more fuzzy and a precise definition of its' usage is elusive [27]. Dingsøy et al. summarize that agile methods emphasize change tolerance, evolutionary delivery, and active end-user involvement [32]. Furthermore, they believe that information systems can be produced with continuous design and iterative improvement, based on rapid testing [31]. Wang et al. determine that the common denominators of agile methods are i. e. short iterative life cycles, quick and frequent feedback from customers and constant learning [122]. Research on the effects of agile observed impacts as a higher job satisfaction [113] or the reduction of bugs [75].

One of the criticism on research on agile development methods is the absence of proper evaluations of actual usage of agile methods [114] and only meager scientific evidence for claims within the agile community [34]. Just because a team declares

that they use agile is not a sufficient enough indicator, because actual usage has often be observed to be lower than claimed [28]. Tripp and his colleagues further criticize that there would have been many anecdotal claims surrounding the benefits of agile vs traditional methods, but that "there remains very little peer-reviewed, published, empirical evidence supporting these claims and substantiating measurable outcomes of agile vs traditional approaches" [114].

2.4.2. Large-scale agile software development

The success and widespread use of agile methods in small, co-located teams encouraged the use in new domains and the increasing utilization of agile methods in large-scale projects [31]. Dingsøy et al. define large-scale development as a development effort that "has more than two teams" and very large-scale "agile development efforts with more than ten teams" [31, 30]. Furthermore, a differentiation can be made between large-scale agile development and "enterprise agile", which refers to applying agile methodologies, principles and values to the whole enterprise and not only software development endeavors [13, 31].

Agile methodologies are not inherently unusable or worse than traditional approaches (e. g. the Waterfall methodology or the V-model) in a large-scale development context, even the opposite can be the case: In a study by Petersen and Wohlin [91], the researchers observed a decline in issues compared to traditional development in a product involving 117 project members, developing three large subsystems in small groups. Another case study indicates that the application of agile principles leads to higher project visibility, improved knowledge-sharing and better coordination on a large-scale [62]. On the other side, there is a multitude of reported limitations and challenges when trying to scale agile methods. A study of three large-scale development cases points out serious challenges, especially the lack of guidance in agile methods regarding dependencies between teams and inter-team coordination [120]. Having multiple agile teams working towards a common goal necessitates a lot of coordinating management effort [91] and relying on emerging architecture could impede project progress [35]. Another interesting aspect, especially relevant for this thesis, is the self-management and self-responsibility of teams. According to Ingvaldsen and Rolfson, self-management of autonomous work groups can significantly impact effective inter-group coordination, which becomes a major challenge in autonomous working teams compared to traditional hierarchical control [51]. In large-scale agile software development, expertise might also be scattered across multiple locations and teams, making networking and knowledge-sharing between teams critical [81]. Various mechanisms are able to support the development of knowledge networks, e. g. collaborating closely with experts outside an individual team, facilitating communities of practice

and providing suitable communication infrastructure [107]. Agile methods are especially problematic for large-scale system architectures and for systems incorporating existent and possibly evolving software architectures [10] or safety critical systems [115]. Another common pitfall is the lack of architecture and unclear role of architects [9]. According to Leffingwell and his colleagues, "some amount of architectural planning and governance is necessary to reliably produce and maintain such systems. Individuals teams, products and programs may not even have the visibility necessary to see how the larger, enterprise system needs to evolve." [67].

2.4.3. Lean software development

The "lean" concept, which emerged mainly from the Japanese automotive industry [122], focuses primarily on maximizing value and minimizing waste in production processes [77]. Another aspect and sometimes called "the essence of the Toyota production system" is that each of the individual employees are given the opportunity to find problems in their own way of working, to solve them, and to make improvements [77]. One example for one of the most commonly used methodologies in software engineering that originates from a lean context are "kanban boards" and related concepts and practices, e. g. set work in progress limits [122]. While there is no uniformly agreed on definition of "lean" in the area of software development and some software literature even see agile and lean "as just two different names for the same thing" [122], a possible distinction can be made that agile methods are more tactical in nature, whereas lean principles and approaches are more strategic and can be applied to a wide scope of different contexts [122]. This differentiation is described in more detail in the next subsection.

Lean is of importance for this thesis not only because of the concept of "lean governance", as detailed earlier, but also because lean is seen as "both the precursor and future of agile", potentially helping to scale up agile practices [121].

2.4.4. Agile vs. lean software development

Wang et al. [122] note that in some examples of software literature, agile and lean are just two different words for describing the same phenomenon. Nevertheless, most literature does consider the difference between agile and lean approaches. One view is that lean and agile are at different levels, with lean being more of a set of principles and guiding ideas, and agile being more at a practice level. Therefore, lean principles could be transformed into agile practices, tailored to individual domain requirements [121]. Another view is that lean and agile are not at different levels but have different scopes and focus areas [121].

In practice, creating and using hybrid methodologies is very common. Especially software-intensive companies tend to select those aspects from agile and lean methodologies that seem to suit them best, which creates new interpretations of agile and lean methods [98].

We believe that it can be important in certain situations to precisely differentiate between agile and lean principles and values because of their clearly different backgrounds and origins. Nevertheless, due to the large number of hybrid applications of lean and agile, especially also in the context of IT governance, as shown earlier in Section 2.1.2, it is appropriate for this thesis to refer to lean and agile as a common theme.

2.5. The interplay between EAM and large-scale agile software development

In the collaborative approach for establishing architecture principles and guidelines presented in this thesis, we combine aspects and insights from different research fields. In particular, those areas are mainly enterprise architecture management, IT governance and large-scale agile software development. Therefore, it is important to build on top of existing research on combining those areas and identifying intersections where these areas can benefit from each other, instead of arbitrarily throwing them together when they might not even profit from each other or where they possibly not be compatible. For that reason, in this section, we analyze the relationship between enterprise architecture management and large-scale agile software development.

The combination of enterprise architecture management and agile software development has been barely researched until now [21, 42, 22], even though both agile development and enterprise architecture are often used in large organizations [22]. Usually, agile software development is not specifically considering enterprise architecture management and vice versa [64, 42]. Hanschke et al. point out differences and similarities between enterprise architecture management and agile software development in their research on how to integrate the two fields [42]. According to them, enterprise architecture management takes a more top-down perspective, in tight coordination with business goals and strategy. It is focusing on setting long-term goals and planning on how to reach them. On the other hand, agile software development is more likely found on a project level. It takes a bottom-up perspective with a more short-term planning horizon [42]. This is consistent with the results of another study on combining enterprise architecture management and agile software development by Canat et al. [22]. All twelve interviewed experts agree on the possibility to combine enterprise architecture management and agile software development, but also stress

that the different levels the two fields operate have to be taken in account. Enterprise architecture is regarded useful mainly on the higher, more strategic level, whereas agile methods are seen as useful in the lower levels, e. g. development and technology [22]. Nevertheless, this gap between the different levels is seen as too large: The lack of communication and the need that enterprise architects and developers work together more closely is mentioned as a key challenge by a majority of interviewees [22].

Hanschke et al. [42] explain why the combination of both enterprise architecture management and agile software development can be very fruitful. Agile software development teams often lose sight of overarching goals and enterprise-wide objectives and focus more on achieving a team-specific, local optimum. Here, agile software development can profit from enterprise architecture management. On the other side, enterprise architecture management can benefit from becoming more flexible and dynamic, as well as from developing a stronger focus on collaboration [42].

Another important aspect to consider when aiming to combine enterprise architecture management and agile development are scaling agile frameworks. The mentioned observation that agile software development is not considering enterprise architecture management and vice-versa is no longer true when taking in account newer frameworks and methods which focus on large-scale agile development, often categorized as "scaling agile frameworks". Since, in this thesis, we put our main focus on large-scale agile development instead of the processes within a single agile team, it is worth to take those scaling agile frameworks into consideration. To further investigate this topic, the research of Uludağ et al., who look into the role of (enterprise) architecture and architects in large-scale agile frameworks, is of particular value. In their research, they demonstrate that three out of twenty analyzed large-scale agile frameworks include the role of the enterprise architect. Even if this is a rather small portion, making up 15% of the total amount of analyzed agile frameworks, two of the agile frameworks incorporating enterprise architecture belong to the three most important ones, in terms of the maturity assessment made by Uludağ et al. [117]. Namely, those are the "Scaled Agile Framework" (SAFe) [104] and the "Disciplined Agile 2.0" (DA 2.0) [5] framework. This further hints to the growing importance of combining large-scale agile and enterprise architecture. The creators of the Disciplined Agile even describe the advantages of using enterprise architecture in agile software development in more detail. They state that enterprise architecture enables reuse across delivery teams. With the possibility of reusing high-quality assets, teams can focus on creating new value for their stakeholders without reinventing the wheel [6]. The common technical guidance that enterprise architecture can provide enables better consistency between teams, resulting in higher overall quality and easier collaboration and exchange between teams. Furthermore, a common infrastructure enables continuous delivery. They conclude that enterprise architecture enables organizations to scale agile strategies across their entire

IT department [6].

2.6. Gamification and social design

Gamification is a rather recent approach, which has been successfully applied in fields like education and health, but is still only rarely appearing in the area of software development and related disciplines [74]. However, there is already existing evidence indicating that such an approach can lead to higher team motivation [45]. Since the focus of our thesis on a more liberal form of governance, including self-governance, is heavily dependant on the team motivation and relying on the involvement of teams, we believe that it is valuable to use insights from gamification and social science to further foster and ensure a high motivation and contribution of teams and individuals. An alternative term for "gamification" proposed by Chou is "human-focused design", which is a design process that incorporates human motivation in systems [24]. It revolves around optimizing the feelings, motivation and the engagement of users and not only on the functions of a system. This means that the key question in the design process is not only "What can the user do?", but "Why would the user do it?" [24]. Chou points out eight core drivers which are crucial and should be taken in account during gamification efforts, e. g. the feeling of development and accomplishment, ownership and possession as well as social influence and relatedness. In addition to gamification elements, it is possible to encourage desirable behavior in communities through certain design principles based on research in social science and psychology [96]. The detailed design principles with relevance and application in this thesis are described in detail in Section 6.4.

3. Related work

This chapter presents and summarizes key publications with significant relevance for this thesis in the areas of enterprise architecture, principles, guidelines as well as agile IT governance and large-scale agile development. The presented related work further extends and elaborates on the foundations presented in the last chapter.

Canat et al. (2018) and Hanschke et al. (2015)

Both Canat et al. [22] and Hanschke et al. [42] are some of the few researchers who investigate the question of how enterprise architecture and agile software development can be combined. Canat et al. [22] study the interplay between enterprise architecture and agile development in their paper titled "Enterprise Architecture and Agile Development - Friends or Foes?". Based on twelve qualitative interviews with professionals in architecture, developer and related roles from five different companies, they conclude that agile development and enterprise architecture can be indeed combined. One of the key findings is that communication between enterprise architects and developers is a common issue. This could be due to the distance between an enterprise architect and the developers. This may also be further affected by a lack of trust and understanding of each other's work. They conclude that both parties would benefit from working together more closely and that it is important to shorten the distance between developers and architects, which fits well to the collaborative approach presented in this thesis. Canat et al. [22] also cite Hanschke et al. [42] as the only other source they could identify that is aiming to answer the question of how enterprise architects can collaborate with agile teams as well. In their paper titled "Integrating Agile Software Development and Enterprise Architecture Management" [42], Hanschke et al. analyze how enterprise architects can collaborate with agile software development teams. Since the combination of agile software development and enterprise architecture management has been barely researched so far, but is a central aspect in this thesis, the paper summarizes a couple of interesting aspects regarding the integration of enterprise architecture management and agile software development. Specifically, Hanschke et al. explore how enterprise architecture and agile development can be integrated by combining a widespread artifact from each side, namely TOGAF [112] and Scrum [105]. For this purpose, they have to deal with "the divergent focuses of ASD and EAM, e.g.

short vs. long-term goals, bottom-up vs. top-down perspectives, single system vs. system landscape" [42].

Dikert et al. (2016)

Dikert et al. [29] conduct a systematic literature review to identify challenges and success factors for large-scale agile transformations. For this purpose, they analyze 52 papers describing 42 different organizations. 46 out of these papers are experience report, which means that the identified success factors and challenges are mainly those that practitioners perceive and declare as important. Some of the identified success factors are management support, communication and transparency, mindset and alignment as well as team autonomy, allowing teams to self-organize. Challenges, among others, include a general resistance to change as well as the coordination of multiple teams in a large-scale environment.

Dreesen and Schmid (2018)

Finding the right balance between top-down oriented governance approaches and self-governance is an important task for the solution artifacts of this thesis. Hence, the question of control vs. autonomy in agile development teams is highly relevant for the scope of this thesis. Because of the lack of existing research in this area, Dreesen and Schmid make a first effort to address this question in their paper titled "Do As You Want Or Do As You Are Told? Control vs. Autonomy in Agile Software Development" [33]. Agile methodologies have in common that they highlight the significance of empowering teams to make decisions, while the management role is focusing more on supporting teams than directing teams. Only limited guidance exists on how agile software development teams should be governed, especially in regards to the mentioned relationship between control and autonomy. Dreesen and Schmid [33] summarize various formal and informal control modes and map them to control mechanisms in agile software development. Those informal control types include clan control and self-control, which fit well to agile values and principles. Nevertheless, the authors conclude that it is beneficial to combine agile methodologies with formal control rather than exclusively with informal control and that their research suggests that agile software development can be flexible and controlled at the same time, resulting in a higher efficiency.

Greefhorst and Proper (2011)

In their book "Architecture Principles - The Cornerstones of Enterprise Architecture", Greefhorst and Proper [37] dive deep into architecture principles and surrounding

relevant concepts and processes. Next to describing the role of enterprise architecture and providing a conceptual framework for principles, they give comprehensive insights into how exactly architecture principles can be specified, with various suggestions for dimensions and attributes that can be used during the specification. They also provide valuable insight by providing a catalog of possible architecture principle examples. Particularly relevant for this thesis is their practical approach, describing "a method and techniques to define and apply architecture principles" [37]. The generic process included in the practical approach is summarized in more detail in Section 2.3.4 within the foundations chapter of this thesis. A quick overview of the creation and management process of architecture principles can be found in the paper "A Practical Approach to the Formulation and Use of Architecture Principles" [38], where Greefhorst and Proper summarize the approach and concomitant process for the development and use of architecture principles from their book.

In their future work section of their book, they state that "it is of the utmost importance that principles are formulated in a collaborative process involving all key stakeholders. More research is needed in effective ways to organize these collaborative processes" [37]. This thesis builds onto this premise and extends the existing approach and process by collaborative traits.

Leclercq-Vandelannoitte and Emmanuel (2018)

Leclercq-Vandelannoitte and Emmanuel [66] apply analogical reasoning to transfer theory, knowledge and experience from philosophy, political and social science to IT-governance. Using the concept of governmentality, first developed by the French philosopher Michel Foucault, they conceptualize a liberal model of IT governance, aiming to potentially replace a more traditional, sovereign IT governance model, based on centralized authority and top-down coercive mechanisms. The liberal IT governance approach by Leclercq-Vandelannoitte and Emmanuel is described mainly in the context and from the perspective of IT usage and not software development. Nevertheless, we see a large overlap with the key principles and values of agile and lean software development methods and advocate the adaption of a more liberal IT-governance approach, thereby making a much better fit in agile and lean environments. Since the implications of a liberal IT governance model are broad, eventually even challenging the very nature and meaning of "being an employee" [66], transforming from being a mere executor of orders to a self-responsible being, liberal IT governance must be understood in a broader context. Therefore, we believe that our approach greatly profits from taking into account these thoughts from a broader context.

The main findings of their work that we consider relevant to the subject of this thesis are the following:

- The role of a liberal governance is to analyze the conditions in which particular behavior occurs and then change the conditions in a way that promotes the desirable behavior [66].
- The question is not how to govern more, but how to govern less and "reach a good balance between the costs of enforcement and the costs of nonconformity" [66].
- Rather than directly enforcing guidelines, help users to make the best choice, by supporting and training them to manage their freedom and educate them about consequences for themselves and the organization [25, 66].
- Greater autonomy and freedom of choice go hand in hand with increased responsibility, accountability and duties [66].

We keep these key findings in mind during the creation of our collaborative approach, which can be seen as an approach implementing a more liberal form of IT governance, due to its' collaborative nature, strongly involving agile teams, resulting in supported self-governance.

Newman (2015)

Newman, the author of the book "Building Microservices" [82], dedicates one chapter of his book on the "evolutionary architect". Whereas the remainder of his book is, as the title suggests, focused on microservices and therefore not relevant for this thesis, the chapter on the "evolutionary architect" describes an approach with many parallels to the approach presented in this thesis. The book is less scientific and is mainly based on practical experience, which makes it an interesting addition to the scientific publications cited in this thesis. Next to a critical examination of the term "architect" and why the term "city planner" offers a better analogy, Newman proposes to guide decision-making by defining a set of principles and practices. These principles should be derived from strategic goals and are intended to align decisions and activities with a larger goal. The practices should ensure how principles are being carried out. He describes them as a set of detailed, practical guidance on how to achieve a certain principle, which is comparable with our interpretation of guidelines. He also mentions the need for finding the balance between autonomy without losing sight of the bigger picture. In line with Greefhorst and Proper [37], he also believes that it is important to document exceptions from the defined principles and practices for future reference. If exceptions have occurred several times, it may make sense to adjust the underlying principles, and thereby to ensure that the principles properly match the challenges that developers are facing. In terms of governance, he suggests a group responsible for discussing and changing principles and practices as required, in structured regular meetings.

3. *Related work*

This group should primarily consist of people who are performing the work that is being governed, that means the developers themselves. He suggests that an architect chairs the group and makes sure that the group works, but the group is responsible for the actual governance decisions. This suggestion is very close to our proposal of a community of practice including enterprise architect and agile team representatives as part of our collaborative approach and it is a positive indicator that our collaborative approach makes sense from a practical perspective as well.

4. Case study

This chapter presents the case study that was conducted during this thesis. First, we present the case study design in Section 4.1, including the objectives of the study, an introductory description of the case, the research questions and a description of the data collection methods. After a case description in Section 4.2, we describe the results based on the four analyzed areas defined by the case study research questions. At the end of this chapter in Section 4.7, we summarize the identified challenges that we subsequently take into account during the development of the resulting solution artifacts of this thesis.

4.1. Case study design

As explained earlier in our approach in Section 1.3, we align our case study with the proposed case study procedure and design by Runeson and Höst [102]. In alignment with their case study guidelines, we describe the case study in the following based on the essential elements that should be included in a case study plan [102]. This is done in order to cover as much of the important meta-information as possible that is essential for the reader of a case study.

Objective — what to achieve?

The case study is of an exploratory nature and the key objective is to identify best practices and challenges regarding governance and enterprise architecture in large-scale agile software development. The focus is particularly on the use of principles and guidelines, as well as the collaboration between enterprise architects and agile teams. The underlying purpose of identifying best practices and challenges is the intent to use those valuable insights, together with the findings from existing research, to create the solution artifacts of this thesis and thereby to ensure practical relevance and value of those resulting solution artifacts.

The case — what is studied?

The case under investigation is one of the largest insurance companies in the world. More specifically, the unit of analysis are the departments, teams and roles that are in touch with the relevant concepts of this research, namely enterprise architecture, large-scale development and architecture principles and guidelines. In the analyzed case, this is mainly the enterprise architecture and IT-strategy department as well as the agile development teams. An insurance company makes a suitable and interesting study target because of the high levels of regulatory pressure that insurance companies face as part of the financial sector. Section 4.2 provides a more detailed description of the case.

Theory — frame of reference

The theory or "frame of reference" [102], in our case the current state of the literature on the relevant topics, is described in detail in the foundations and related work in Chapter 2 and Chapter 3.

Research questions — what to know?

The research questions set out to be answered during the case study are the following:

1. How are architecture principles and guidelines used in an agile environment?
2. Which architecture communities are established in the company?
3. What is the role of the enterprise architects in an agile environment and how do they collaborate with agile teams?
4. What value do the enterprise architects create in an agile environment?

These questions are also reflected by the four questionnaires used for the semi-structured interviews that are part of the case study. The questionnaires can be found in Appendix A.2. Analyzing these questions has the goal to receive relevant background information and a better understanding of the situation and opinions in the case study company. Those findings then can provide valuable insight for answering the overall research questions of this thesis, which are outlined in Section 1.2. As explained in our approach in Section 1.3, we also use the environment of the case organization as part of our action research to continuously demonstrate and evaluate our prototypical implementation, which is described in more detail in Chapter 6. The more comprehensive, formal evaluation is also done within the scope of the case study company. This evaluation is described in more detail in Chapter 7.

Methods — how to collect data?

Due to the importance of using several data sources in a case study and taking different viewpoints and roles into account [102], we use the following methods to collect data:

- Unstructured interviews
- Semi-structured interviews
- Observations
- Document analysis

We used informal, **unstructured interviews** on an almost daily to weekly basis during our analysis at the case study company, especially for further investigating and understanding the problem domain, as well as refining the requirements used during the artifact development. As explained earlier, those continuous activities were of special importance in this thesis, to ensure ongoing adjustment and continuous improvement of the artifacts in development, based on new insights and feedback. The main roles that we communicated and collaborated with were enterprise architects with agile software development experience. We have also conducted unstructured interviews with members of agile teams several times, but less frequently than with enterprise architects due to availability reasons. In addition, we conducted formal, **semi-structured interviews** with key stakeholders to achieve more systematically collected and analyzable data as well. The main difference in semi-structured interviews is that all questions are planned in advance, containing a combination of open and closed questions [102]. The question catalog was developed collaboratively with other researchers, to achieve a homogeneous list of questions that are used not only in the present thesis but also in case studies with other industry partners. This ensures that the results from the various case studies can be aggregated and analyzed for more meaningful insights later on in further research. These semi-structured interviews were conducted with five interviewees in four interviews. The participating roles include two enterprise architects, one of whom is more senior, whereas the other one has only recently switched his role from an agile developer and software architect to enterprise architect, while still carries out some development and software architecture tasks. Furthermore, two agile developers were interviewed, with one of the developers having an additional role which is similar to the role of a scrum master in the scrum methodology [105]. The fifth participant is an agile manager responsible for all scrum masters of the agile development teams in the case study organization. Due to the rather long planned time frame of the interviews (around two to three hours), the interviews were split into multiple sessions to account for the tight schedules of the

interviewees. Table 4.1 shows the participants of the semi-structured interviews with their respective alias, which we use in Sections 4.5 and 4.6 to map statements to the respective role that they originate from. The alias of the first and second interviewed agile developer is combined because of their very strong overlap and agreement in statements and opinions during a joint interview. The question catalogue of the semi-structured interviews can be found in Appendix A.2.

No.	Main role	Alias	Professional experience in that role
1	Enterprise Architect	EA1	3-5 years
2	Enterprise Architect	EA2	3-5 years
3	Agile Developer	AD	2-3 years
4	Agile Developer	AD	6-10 years
5	Scrum Master Lead	SML	3-5 years

Table 4.1.: Interview partners of semi-structured case study interviews

To further enrich our data collection, we conducted **document analysis** of relevant documents identified in the target organization and **observation** by participating in various meetings on topics and areas relevant to the scope of this thesis.

As described in our approach in Section 1.3, the case study was not limited to a short time frame, but ongoing throughout the whole time scope of the thesis. This allowed for continuous analysis, feedback and validation of results, especially also for the development of the prototypical implementation of the web-based application presented in Chapter 6. With regards to the design science process, the case study is used to enrich all phases described in the approach in Section 1.3, but also used for the evaluation, which is described in Chapter 7.

The following sections summarize the results from the semi-structured interviews, informal interviews, document analysis and observations at the partner company, facilitating a better overall understanding of the status quo at the company.

4.2. Case description

The case under investigation is a large insurance enterprise. To keep pace with rising customer expectations and rapidly changing environments, the organization decided to start using agile development methods over two years ago in 2016. This was done by setting up dedicated locations with co-located teams, detached from the traditional development endeavors. In these dedicated locations, the employees of the company are being trained in agile development methods and develop products and services with early consideration of customer feedback. The co-located agile teams are composed

of experts from various fields, including developers, designers as well as product and process experts. One of the goals of this collaboration is to achieve more customer-oriented and faster results. Within the agile environment, the company relies heavily on the use of platform as a service (PaaS) products, integrated build pipelines with automated testing and in general, closely following the cloud native [111] ecosystem and community. Despite the modern work environment and the attempt to keep the agile teams separate from the traditional development efforts of the company, there are still certain structures, regulations and existing processes that impact the agile development teams in their daily work.

4.3. Use of architecture principles and guidelines

This section described the findings regarding the use of architecture principles and guidelines within the agile development environments of the analyzed company, answering the first research question of the case study.

Finding a clear answer to the question of which principles and guidelines agile teams should adhere to and how exactly principles and guidelines are used in the organization has proved difficult. There are several reasons for this, which are explained in more detail in the following. Since the organization mainly uses the term "guideline", the following description will also mainly rely on this term. First of all, there is the fact that, due to the large size of the company, multiple stakeholders scattered over different departments have an interest in influencing the agile development teams. This results in different sets of principles and guidelines.

Examples for these departments and sets of guidelines include an operations department that is providing the infrastructure and services to run the developed applications. They have a legitimate interest in providing guidelines for teams on how to build and deploy applications in a way that ensures low-costs and high stability. Furthermore, there is a separate department that centers on software quality assurance, which also communicates with departments that focus on legal aspects. Guidelines published by the software quality department mainly have the purpose to ensure that teams fulfill legal requirements, which demand certain forms and levels of documentation, traceability of changes to productively deployed applications as well as a safe, longstanding audit-compliant storage of those documents and artifacts. Because the company is active in the financial sector, there are specific, more strict national regulations for IT-usage that have to be taken into account. This is also reflected by the existence of a dedicated IT-security and data protection department, which also sets certain rules for developers.

The enterprise architecture management department provides a set of guidelines as

well. There is a precisely defined and well documented process on how new guidelines can be created. The process even already has collaborative aspects. It allows and encourages everyone to suggest guidelines to the appropriate architecture committee or send them directly via e-mail to a dedicated guideline team. The members of the guideline team then assist the guideline proposer with the next steps, especially with identifying the suitable responsible architecture committee based on the type and scope of the new guideline. There is an available template that suggests certain attributes for specifying guidelines as well as categorizing them into different levels of architecture. The predefined levels of architecture are business architecture, product architecture, data architecture, service architecture, technology architecture, application architecture, and software architecture. The guideline proposal may then be submitted to the responsible architecture committee. After an initial approval of the proposal by the committee or a further iteration through a chief architect, in case of doubts regarding the proposed guideline within the committee, the proposal gets refined further and is then submitted to a "request for comment" phase. During this phase, the proposed guideline gets published in a wiki and community platform where other architects and interested employees are now invited to provide their feedback within a fixed time frame. After this phase is finished, the responsible architecture committee makes a final decision whether to accept or decline the proposed guideline, also taking in account the collected feedback. When the decision to accept the guideline is made, the new guideline is published on a dedicated wiki page, which houses all guidelines from the enterprise architecture department. Afterward, small changes can be made self-responsibly or with the help of the guideline team, as long as the committee is informed of the change. More substantial changes to a guideline require the creation of a new guideline.

One of the problems is that, while the categorization in different architectural levels provides a good way to structure and easily navigate the existing guidelines, it lacks information on which guidelines have which target groups. As explained earlier, the company is not relying solely on agile development yet but uses non-agile development methods especially in the development of existing core applications. Therefore, many of the guidelines aim at areas that are currently not relevant to agile teams. This makes it difficult for agile teams to identify guidelines that are useful and important for them, provided that the teams are even aware of the existence of these guidelines at all. The broad possible scope of guidelines may also hinder participation in the community. Since guidelines and guideline proposals are not published with information on the intended target group, it may be hinder participants to quickly identify that there are relevant guidelines in their area of expertise that they can contribute to. The more significant problem, however, is the lack of an overall guideline process that also focuses on other important areas besides the creation of guidelines, e. g. applying guidelines,

managing their compliance or more details on handling change.

The fact that there are no defined measures for applying principles or managing their compliance severely impacts the degree that teams are aware of those principles and guidelines. While the architecture boards are taking existing guidelines into account when reviewing architectures of certain teams, the participation and presentation in this review process is voluntary for agile teams. A common theme that was mentioned during unstructured interviews is that the lack of consequences for not complying with guidelines is leading to a lower level of compliance. This is further impacted by the lack of transparency as to which teams actually apply which guidelines. Additionally to the department-specific guidelines, there are also guidelines defined by some individual projects, e. g. a project to migrate legacy applications to a cloud platform. On top of that, there are an increasing influence from the superordinate holding company with the goal of harmonizing the architecture globally over multiple national operating entities. They also suggest certain principles and guidelines as part of a global blueprint that should be taken in account by the organization and might be enforced more strictly in the future.

Another issue, in addition to the existence of multiple different sets of guidelines and involved stakeholders, is the lack of a joint, uniform communication and documentation of principles and guidelines. Teams have to communicate with multiple stakeholders based on the area of principles and guidelines that they are mainly responsible for. It is often unclear which guidelines have to be used under which conditions and which guidelines are more best practices than strict rules. Furthermore, there were cases where it was difficult for teams to identify the proper contact for certain guidelines or it was unclear where guidelines were coming from and what the underlying rationale and reasons for the guideline are. Also, regarding guidelines from the enterprise architecture department, there is limited participation in the community described earlier. The community is especially limited in diversity, meaning that architects are more likely to participate because the group was previously limited to architecture roles, but is now open for everyone who is interested. But awareness and participation of agile development teams are still low. This issue is further aggravated by the fact that there is no central guideline repository that covers all relevant guidelines, or a single entity or group that makes sure that guidelines from different stakeholders are merged, taking in account inter-dependencies and overlaps between various guidelines to harmonize governance efforts.

Nevertheless, the observed status quo is subject to continuous improvement. One example is the planned introduction of architecture training for every agile team as part of the onboarding process, with the goal to sharpen awareness regarding architectural topics within agile development teams. When the first agile teams were established, there was an emphasis on not restricting those teams at all and the decision was

made that these teams do not have to adhere to guidelines published by the enterprise architecture department. But over time it got clear that some sort of guidelines are required, especially in recent months because of the growing importance of global harmonization efforts and the goal to standardize and reuse assets globally throughout different operating entities.

4.4. Architecture communities

This section gives an overview of the different architecture communities that are present within the company, in regards to the second research question of the case study. As briefly mentioned in the previous section, there are various architecture committees overseeing the architectural decisions individual projects make. These committees are more traditional in their organization, meaning that they have fixed members and chairmen, and they act on different levels and topics. On the organizational level, there is a committee that focuses on strategic global decisions. This involves investments or investment stops for certain technologies, frameworks or software. They also set certain standard software, directions for innovation and global infrastructure platforms. The committee on the level of the local IT-organization focuses on IT standards based on functional domains and propose technical reference models for each domain. On the portfolio level, there is an architecture committee that looks more closely into projects and the fit with architectural requirements. On this level, there are additional, separate architecture committees dedicated to services, data architecture as well as customer interaction. Agile teams are not forced to present in those architectural boards, but they are encouraged to seize the opportunity to validate their architectural compliance and get support and guidance on pressing issues that their team faces. On a program and team level, chief product owners and product owners make the decisions together with the agile development teams, which, in the usual case, always consist of multiple developers with at least one developer taking on the additional role of a software architect. Within the agile development community, there are various communities of practices (CoP), with focus on certain topics, e. g. for scrum masters (or at least a role similar to a scrum master in Scrum that is used in the company) or DevOps enthusiasts. Some of the enterprise architects have recently started joining these communities on a regular basis as well, achieving a closer collaboration with agile teams.

4.5. Role of the enterprise architect

The following section describes the analysis of the third research question of the case study, the role of the enterprise architect within the agile environment, in more detail.

The italic aliases in parentheses after certain statements indicate the role behind the statement and refer to the aliases listed in Table 4.1.

Enterprise architects in the partner company have three core responsibilities: Architecture governance, incubation and enabling. Architecture governance revolves around architectural specifications, the organization of the architecture committee, creation of guidelines and architectural models and blueprints of the target landscape. Incubation targets the evaluation of new technologies, whereas enabling has the goal to provide reusable assets to simplify the work of agile teams and support their day to day efforts, e. g. by providing a standardized continuous deployment pipeline and cloud platform. Furthermore, enabling includes training and coaching to teach the most important architecture basics. In terms of responsibility, the goal of the enterprise architecture in the organization is that they do not have to participate and approve every single decision, but that they let agile teams make the decisions and let them take over responsibility (*EA1*). This requires the enterprise architects to ensure that existing guidelines are of high quality and, above all, that teams are aware of them (*EA1*). The agile developers see the core responsibility of enterprise architects in communicating important information to the teams, collaborate closely with the teams, provide access to their network, identify necessary contacts and support with building common assets (*AD*). These may include development platforms or continuous deployment pipelines, enabling the teams to work more quickly (*AD*). In their view, the responsibility of enterprise architects has decreased in the agile context because teams are more self-responsible, also requiring enterprise architects to be willing to give up some of their power (*AD*). On the other side, they stress that more effort and commitment is now required to be able to explain and communicate existing guidelines, because instead of relying solely on command and control, they now have to achieve acceptance on the part of the teams and provide real value for the teams (*AD*). Both interviewed enterprise architects agree with this view and stress that enterprise architects have to generate tangible added value within the project teams (*EA1*) and have to clearly communicate what they do and how they can support agile teams (*EA2*). This is required so that these teams approach enterprise architects on their own initiative to work together (*EA1*, *EA2*) and that enterprise architects also get invited into collaborating more closely with agile teams, e. g. in CoPs that are mainly driven by the agile teams themselves (*AD*). Enterprise architects are also responsible for distributing knowledge across the teams and help them to get started quickly when approaching a new project or product (*SML*).

Some of the enterprise architects of the organization are already working in a more collaborative way with agile development teams than traditional architects. Instead of being isolated in their "ivory tower", they work directly with the teams and are even part of those agile teams for a certain time period to support them. This is

something that both enterprise architects and agile developers call for. According to the developers, enterprise architects should not be a permanent part of a fixed agile team, but they should rotate through teams (*AD*). *SML* also stresses that enterprise architects are no longer in a position to only create rules, but are a kind of service provider who can provide a certain architecture and support, and who is also responsible for communicating the architecture and its rationale.

Enterprise architects in the company already put emphasis on direct communication and try to actively approach teams and provide them with information. In line with the wide range of different enterprise architecture activities in the organization, the team of enterprise architects is very interdisciplinary, consisting of experts with different technical expertise and skills. Within the agile environment, the working methodology changed in the sense that the direct collaboration and information sharing with agile development teams became significantly more important (*EA1, EA2, AD, SML*). This is also reflected in the relevance of different tools. Previously, there was a higher focus on modeling in specialized tools. Nowadays, the focus is more on collaboration tools (Wiki, issue trackers, community platform etc.). Enterprise architects should also work in a more agile way themselves, conducting weekly stand-ups and using backlogs and Kanban boards (*EA1*) to distribute and manage tasks, which is already partly the case in the organization. Enterprise architects have a lot of self-responsibility and can take initiatives on their own to suggest new projects or directions they would like to head to.

All interviewed stakeholders see the need and value in enterprise architects joining teams for actual coding (*EA1, EA2, AD, SML*), which might not directly benefit the enterprise architecture (*EA1*), but strongly benefits the acceptance by agile teams, which is the basis for everything else (*EA1*). Therefore, the approach of enterprise architects is less-top down driven, but becoming more collaborative (*SML*). Nevertheless, especially in large organizations as the case study company, traditional enterprise architects who work mainly strategically are still import and should not and cannot be part of agile teams (*EA1*).

Both enterprise architects see the future focal point of enterprise architecture in automation. Architectural documentation could be increasingly automated (*EA1, EA2*), as well as the data collection for certain metrics and KPIs (*EA1, EA2*). Self-service offerings for software development teams to quickly spin up application templates and build pipelines with necessary integration should be provided as well as tools and methods for automatically testing guidelines (*EA1, EA2*). Architecture has to be integrated into modern development and deployment platforms, but more strategically focused enterprise architecture that can not be automated will be still important in the future (*EA1*). One of the interviewed enterprise architects states that the current architecture role might not even be required any more in this particular form, but it might be more in the form of a "community manager"(*EA1*). Because in the ideal

case, architectural artifacts (e. g. blueprints) should originate bottom-up through the agile teams and not top-down (*EA1*). The enterprise architect further sees the vision of establishing architecture as a product (including e. g. a wiki, development platform, continuous delivery pipeline, and enterprise architecture tool) with a chief architect as the product owner and the various agile teams as customers, with more systematic feedback processes and retrospectives (*EA1*). This need for a more systematic and regular feedback cycle between architects and agile teams is also mentioned by the other interviewed stakeholders (*EA2, AD*).

A challenge for the enterprise architects is the unclear role within agile processes, because prevalent agile methodologies do usually not consider the role of an enterprise architect (*EA1*). This requires experimentation, testing and evaluation of what works and what does not work, as well as research and exchange of experience with industry partners, to coin a more concrete role of the enterprise architect in agile settings (*EA1*). Establishing such a new agile enterprise architecture role and process requires a lot of time and resources (*EA1*). This challenge is exacerbated by the quickly changing environments of the enterprise architects, forcing them to rapidly adjust with those changes (*SML*). According to *SML*, this makes enterprise architects that are closely collaborating with agile teams even more important, because they have the necessary proximity to agile teams to quickly recognize changes and adapt accordingly.

Another important challenge is the acceptance of enterprise architects as well as principles and guidelines by agile teams (*EA1, AD*), because teams do not always see enterprise architects as necessary and do not like to be controlled (*AD*). This may be a reason that existing guidelines are often being ignored (*EA2*). The challenge for enterprise architects is how to develop those principles and best practices together with the teams (*AD*). This is made more difficult by the tendency of teams to challenge and question everything that is related to certain predefined standards and guidelines defined by other stakeholders, even if they do not know the details behind the decisions or the evidence that lead to certain decisions (*EA2*). A further challenge is that the responsibility is increasingly passed on to the teams, without that these teams are even aware of all the responsibilities that they now have to take care of (*EA2*). Another common problem mentioned by the interviewees is the lack of technical expertise and proximity to technical implementation of enterprise architects. They interviewees stress that this is not a problem that they see in the case study company, because enterprise architects mostly have sufficient technical backgrounds, but it is a challenge that they have seen or experienced in other companies (*EA1, EA2, AD*).

The interviewees state various recommendations for the role of the enterprise architects in agile environments. Enterprise architects should closely examine modern development platforms, because architecture should no longer be on slides, in documents or wiki pages, but directly integrated into the platform (*EA1*). They should

also try to look at enterprise architecture from the perspective of agile teams and tailor their role accordingly, in a way that enables them to bring value to those teams (*AD*). They should also be present and co-located with the teams (*EA2, AD*) so that they are able to identify what challenges teams face and where the teams could use support (*AD*). This also gives enterprise architects valuable insights into how teams work, how decisions are made in teams and what consequences decisions by other stakeholders have for the teams (*EA2*). Furthermore, *SML* recommends that enterprise architects focus on providing value for the teams and on their ability to communicate and present their concepts. The generated value should be measured based on the needs of the organization (*SML*).

4.6. Value contribution of enterprise architects

This section focuses on the fourth research question, the value contribution of enterprise architecture in the agile environment.

As stated previously, enterprise architects cannot rely on command and control anymore, but they have to create value for the agile teams, ideally in a way that the teams ask for more help on their own, because of the advantages they can benefit from (*EA1, EA2, AD*).

Regarding the expectations agile teams have of architectural models and other artifacts delivered by the enterprise architects, all interviewees stress the importance of the reliability and binding nature of the deliverables and information communicated by the enterprise architects (*EA1, EA2, AD, SML*). Teams would like to be able to rely on the information they receive and might refer to it at a later time (*EA2*). They also expect that architectural models are developed collaboratively and are not "carved in stone" (*EA2*), but can be discussed with the architects. The teams also expect architecture models to be up-to-date and relevant for the team, with availability during the project initialization where they are most helpful, because they can be used for solving initial problems (*AD*). They would also like to understand the motivation behind the architectural artifacts and what exact value it can provide for their team (*SML*).

Regarding the question if and how teams would like to be controlled or guided by enterprise architects, the agile developers have the opinion that agile teams do not want to be controlled at all and that exercising control is also not part of the tasks of enterprise architects, because the teams should be self-responsible (*AD*). They do however acknowledge the potential need for a type of "sign off" on go-lives to production or regarding legal requirements, ideally through automated testing (*AD*). The manager, who is responsible for all Scrum Masters in the case study organization, also believes that teams want and should be able to work as autonomously as possible

(SML). Nevertheless, there is a need for central coordination and for someone to provide centralized services. However, this should take the form of a service provider rather than a regulator (SML).

The enterprise architects take a more differentiated view of the situation. From the perspective of developers, there should be no control and the ability and freedom to decide for themselves (EA1). Nevertheless, a product owner could have a high interest in being compliant to certain standards and receive guidance and information for achieving this compliance (EA1). The level of control necessary or desired also depends highly on the individual team. Some teams even approach enterprise architects on their own initiative and ask for guidance, while other teams have a very low acceptance of control or guidance, depending on the individual characters and experience within the team (EA2). Nevertheless, most teams would like to understand beforehand what their scope of action is, as well as what they are free to decide and where they have to take some regulations into account, instead of having to deal with issues after taking a certain course of action (EA2).

Guidelines and standards that would bring value to teams are those that explain legal requirements and best practices for solving them, as well as guidelines and reusable assets on cross-team topics, e. g. logging, monitoring, security, version control (AD) or UI-components, services, development and runtime environments (EA1). Concrete technical guidance, e. g. using predefined technologies and thereby limiting the technology selection of individual teams does not appeal to teams (EA2, AD), but might significantly help the overall organization and enterprise architecture (EA1, EA2). Teams focus more on how they as an individual team can build the quickest, but that does not mean that it is the ideal solution in the long run, e. g. regarding operability and maintainability (EA2). However, it can be a problem if standardization does not allow for enough heterogeneity, e. g. to test and evaluate new technologies (EA1).

4.7. Summarized challenges

While the last sections already provided more detailed information regarding best practices and existing challenges, in this section, we would like to summarize a selection of the main relevant challenges that we identified at the case study company. These challenges are then taken into account during the following chapters. Mainly relevant for the scope of this thesis are challenges related to architecture principles and guidelines as well as the collaboration between enterprise architects and agile teams. Table 4.2 shows an overview of the identified challenges. The collaborative approach presented in Chapter 5 and the tool support introduced in Chapter 6 address these challenges.

ID	Main challenge
C1	No single point of truth for guidelines
C2	Multiple involved stakeholder groups
C3	Lack of awareness and compliance with guidelines
C4	Acceptance of principles and guidelines by agile teams
C5	Unclear implications of guidelines
C6	Lack of relevance and applicability
C7	Lack of transparency
C8	Principles and guidelines require additional time and effort
C9	Architecture integration into the platform
C10	Collaboration and feedback cycles

Table 4.2.: Main relevant challenges identified during the case study

There is no single point of truth or a clear and easy overview of all existing guidelines that a team should adhere to (C1). Guidelines are spread throughout multiple channels (e. g. different wikis) and documented in different forms. There is no single entity which takes care of managing guidelines, but there are multiple stakeholders (C2) who have an interest in defining principles and guidelines, further complicating the situation (e. g. interests from business, infrastructure, architecture or security departments). This makes it difficult for teams to be aware of which guidelines exist and which guidelines are relevant to them, therefore the lack of compliance with guidelines is a challenge (C3). This lack of compliance is further exacerbated by the lack of transparency which guidelines are being fulfilled by which teams (C7) and the lack of consequences for not fulfilling these guidelines. Teams do not always care about guidelines or do sometimes simply not know that they exist. Sometimes they also ignore guidelines by purpose because of a lack of acceptance regarding specific guidelines (C4). This also makes it difficult for the guideline creators to better understand the impact of their guidelines at the implementation level (C5), which could be used to assess whether a guideline is actually valuable and to provide valuable feedback for further refining guidelines. Another one of the main challenges is the lack of relevance and applicability (C6). Not all guidelines are equally relevant for each team. There might be guidelines that are only relevant when certain preconditions are met or when the team is active in a certain functional or technical domain.

The lack of transparency (C7) is also a challenge from the perspective of the teams. For them, it is often not clear why they should comply with a certain guideline and what the rationale for the decisions described in the guideline is. On top of that, it is not always clear to teams whether they have to adhere to a guideline or whether it is more

4. Case study

a best practice that they should follow, but from which they may also deviate. Another major challenge is the additional time and effort required in the area of architecture principles and guidelines, both for principles and guidelines creators as well as agile teams who should comply with those principles and guidelines (C8). On the one hand side, it means effort for agile teams to familiarize themselves with existing guidelines and to ensure the compliance with those guidelines. Furthermore, it can be a lot of effort for those teams to communicate with the involved stakeholders because of the different stakeholders involved (C2), leading to uncertainty whom teams have to contact for questions or problems with certain guidelines or how they can get an exemption if they face special requirements. In the worst case, the progress of a team can even be impeded due to slow governance processes. Teams may have to wait for a decision or architectural approval so that they can go ahead with their planned endeavors. On the other hand, it causes a lot of effort to manage the compliance to guidelines, since the lack of transparency (C7) and the fact that guidelines are not automatically tested. This leads to the challenge on how to integrate the architecture directly into the development platforms (C9), so that the compliance to guidelines can be assessed at least in part automatically. This would further reduce the required manual effort and enhance the awareness and compliance with guidelines. Lastly, we would like to highlight the challenge that enterprise architects and agile teams have to collaborate more closely (C10). The collaboration challenge is not only exclusively relevant for the relationship between enterprise architects and agile teams, but the agile teams would also profit from more inter-team collaboration between each other.

As this brief analysis shows, many of the challenges are mutually reinforcing. Overcoming some of these challenges could therefore also reduce challenges that have not been addressed directly and contribute to solving them.

5. Collaborative approach to establish architecture principles and guidelines

This chapter presents one of the two main artifacts of this thesis: a collaborative approach to establish principles and guidelines that is combining top-down and bottom-up processes. First, in Section 5.1, we provide an introduction for our collaborative approach. Afterwards, we describe the collaborative approach in detail in Section 5.2: The involved stakeholders and their role in the approach, the guideline establishment process as well as the specific activities of the community in the collaborative approach. The final part of this chapter is Section 5.3 where we outline the addressed challenges and resulting solution requirements.

5.1. Agile governance through collaboration

Our approach focuses on achieving agile governance by bringing principles and guidelines, an established artifact from traditional IT-governance and enterprise architecture management, to modern agile and lean environments. As outlined in the foundations in Chapter 2.1.2, our understanding of agile and lean governance is that it revolves around a more liberal stance on governance, putting more trust in agile teams and giving them more responsibility, resulting in a form of self-governance.

Achieving the goal of a collaborative approach requires an adaption of the existing approaches and processes related to principles and guidelines. This adaption is necessary and important because traditional, top-down methods lack the consideration of bottom-up processes and put the main emphasis on top-down, "command and control" methods, which we see as contradicting with agile and lean values.

Scott Ambler, creator of the "Disciplined Agile" framework [5] during his time as chief IT-methodologist at IBM Rational, states that "Traditional IT governance strategies prove to be at odds to agile's collaborative, value-focused strategies. As a result, a lean approach based on enablement, collaboration, and motivation is required to effectively govern agile teams." [7] We regard the consideration of the bottom-up view as an adequate fit with the lean mindset because, as stated by experts on the Toyota production system where lean values originate from, "the essence of the Toyota production system is that each individual employee is given the opportunity to find problems in his own

way of working, to solve them, and to make improvements" [47]. Hamel summarizes Toyota's advantage as the "ability to harness the intellect of 'ordinary' employees" [41].

In our approach, this means that agile teams are no longer only affected by the results of the principles and guidelines governance process, but that they are key stakeholders themselves in determining, specifying, applying and refining principles and guidelines. This approach pushes decisions to the people closest to the implementation, meaning the people who are actually executing the work being governed. This increases both the power and autonomy of the individual teams, however at the same time also making them more accountable for their actions.

To achieve the goal of agile teams becoming a key stakeholder in the creation and application of architecture principles and guidelines, without losing the necessary oversight to achieve organizational, cross-team goals, we propose a community of enterprise architects and agile developers who collaboratively conduct a process for establishing architecture principles and guidelines. For this purpose, we adopt and adapt the generic architecture principle process by Greefhorst and Proper [37] to take the additional perspective of the agile teams into account. Greefhorst and Proper admit that their comprehensive process, covering the whole architecture principle life-cycle, is focusing on a top-down approach and that more work has to be done to include bottom-up perspectives into their approach [37].

And this is precisely where we position our collaborative approach for establishing architecture principles and guidelines: we build on existing, well-established approaches and adapt and enrich them in a way that focuses on collaboration and enablement of affected stakeholders instead of strict control.

The combination of the top-down and bottom-up perspective is what we call the *collaborative* establishment of principles and guidelines. And when we speak of the collaborative *establishment* of principles and guidelines, we mean that we take into account the full life-cycle of principles and guidelines. This means that our approach encompasses not only the creation process itself on a strategic level, but the actual application of principles and guidelines during implementation and handling of change by incorporating experience and feedback.

It is important to note that our approach is not a golden hammer, but has to be adapted to the specific requirements of an organization. As Greefhorst and Proper conclude regarding the architecture principle development process: "It has been shown that the development of architecture principles is very situational. In particular, the type of architecture, the maturity level of the organization and the culture very much influence the approach" [37]. To take this into account, we will mention instances where we see a particularly high need to adjust our approach to the organization maturity and culture. Furthermore, it is worth mentioning that we do not focus on suggesting specific architecture principles and guideline examples. Instead, we aim our attention on the

way how such principles and guidelines can be collaboratively developed. This is due to the fact that concrete principles and guidelines are highly specific to the individual organization. Greefhorst and Proper also acknowledge this by stating the following regarding their catalog of architecture principle examples: "Experienced architects probably do not need the catalogue; they completely depend on their personal instinct, experiences and knowledge" [37]. Besides, the process and the detailed order of steps is not a strict specification, but it should provide an indication of what activities should be done in such a community. The central aspect of the approach is that the individuals from the different stakeholder groups come together and interact more frequently, in line with the first agile value of the agile manifesto [14].

The value of our approach is that, in the ideal case, only the governance remains that is inherently useful to the target stakeholders, without losing the oversight, long-term planning and collective goals of the organization. This also encourages those stakeholders to give valuable feedback on existing guidelines, making it possible to improve them further and adapt them to the needs and challenges that agile teams face on a day-to-day basis. Ultimately, the approach allows companies to decentralize their decision-making and enable teams to manage themselves as much as possible.

5.2. Guideline establishment approach with relevant stakeholders

The following section presents the key stakeholders and their role in our process, the way how they come together to collaborate on architecture principles and guidelines, as well as the steps of that process itself.

5.2.1. Involved stakeholders and their role in the approach

This section describes the central roles that are part of our approach, without detailing their function in each specific step of the process yet.

In our approach, we focus mainly on two roles: The enterprise architect and the agile development teams. Nevertheless, our approach could also be generalized. The two roles could also be described as the group that creates principles and guidelines and the group that has to comply with those principles and guidelines. In the end, the essence is to melt these two groups into one community, thereby combining top-down and bottom-up views within one process.

The traditional approach is shown in the left column of Figure 5.1. Guideline creators are on their own in creating guidelines, which are then communicated to the guideline target groups. A more participatory approach can be seen in the center. Guideline

5. Collaborative approach to establish architecture principles and guidelines

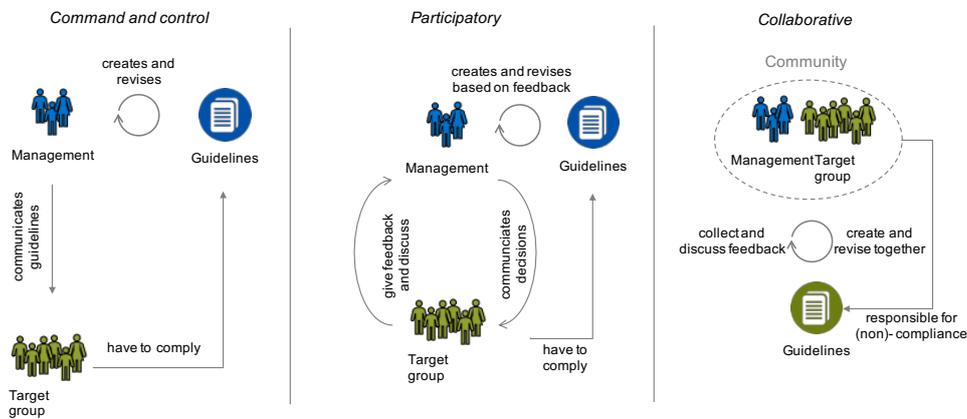


Figure 5.1.: High-level overview of the key difference in the collaborative approach

target groups can now participate in the creation process, e. g. by giving feedback to guidelines. From a collaborative perspective, this is already an improved situation and closer to the collaborative approach than the traditional form. However, the basic outcome is the same: Guidelines are created top-down and teams have to adhere to the guidelines without a lot of self-responsibility. The desired process is illustrated on the right. The crucial difference is that there are no longer separate groups for guideline creators and the group who has to implement the guidelines. Instead, the people who have to implement the actual governance intentions are the ones who shape the guidelines. They can actively join in on the process and the participating stakeholders have a common goal: to establish guidelines that are valuable and relevant to the target groups. However, this does not mean that only the groups who are performing the work being governed are involved in the process. An overarching perspective is still important.

Through the involvement of enterprise architects, the more strategic perspective is included that aims to reach a global, organization-wide optimum. By including agile teams, the operational perspective is included as well. Those individual agile teams are the experts in their project or product scope and can aim for a local optimum. By combining both views, agile teams get more aware of the global optimum as well, and how they have to adapt their goals to not only reach a local optimum for their team, but also conform with the goals of the organization. Even if we speak mostly of developers in the following explanations of our approach, the community could also be interesting for the other possible members of an agile team, which might include business analysts, designers or product experts. If additional stakeholder groups have been involved in the creation of principles and guidelines or other governance measures, they would

also have to be included in the community, or at least have to be taken into account by enterprise architects. The approach also has to be embedded into a broader scope, that means it should have awareness and support of the upper management and they should reward participation within the community appropriately.

When it comes to increasing the autonomy of agile teams and achieving some form of self-governance and self-management, one could argue that agile teams themselves should be the only stakeholders in the process. This would make the role of enterprise architects superfluous. However, we believe that the role remains important. Since self-management can negatively affect the ability to effectively coordinate between teams [51], it is vital to have a role that can assist and support with cross-team coordination. Furthermore, when teams should be able to self-manage in large-scale environments, they need to have a strong knowledge network and collaborate closely with experts outside of their team [31, 81, 107]. Enterprise architects can be of immense value by providing teams with such a network and helping them with identifying the right contact persons in a short amount of time. Moreover, enterprise architects usually have a better insight and understanding of organizational goals and strategy, and thereby a better ability for business and IT alignment. Nevertheless, in our approach, we see enterprise architects in a different role than they might have had before. Architects in some companies are still in a more traditional role: they create diagrams, models, slides, documentation on the perfect system, but do not take in account the unforeseeable future. We see the modern architect in a different role, which could be more accurately compared with a "town planner". The analogy of a "town planner" was first introduced by Doernenburg [82]. As described in more detail by Newman, the town planner uses a myriad of information to attempt to "optimize the layout of a city to best suit the needs of the citizens today" [82]. The way a town planner does this optimization is interesting because it fits well to the general idea of the liberal governance approach: He does not specifically state "build this specific building here". Instead, he splits the city into different zones, e. g. by designating parts of the city as residential zones or industrial zones. It is then the decision of the people what exact buildings are going to be built. The town planner will then primarily focus on what happens between the zones, but not necessarily what happens specifically within a zone [82]. This is very similar to how we position the enterprise architect in our approach. The shift goes away from making detailed decisions himself, but to focus more on supporting and enabling others to make these decisions themselves within certain boundaries, and document resulting insights on the way. The core responsibility of the enterprise architect in our approach is facilitating the collaboration, bringing the community together and supporting them in reaching the community goal. Especially in the initial phase of such a community, it is important that someone takes the initiative and assumes responsibility for establishing the community. To build such

a community, it would be advisable for enterprise architects to look for allies within the agile development teams. This could significantly improve the initial acceptance and prevent the impression on agile teams that this is another initiative that is being forced on them. The biggest obstacle at the beginning of such a community could be that it requires a certain level of participation to create value, but at the same time, it requires a certain level of value to achieve enough participation. The use of the network of enterprise architects to identify enough potential participants for the first steps of the community could be crucial to cross this value and participation threshold.

5.2.2. Community goals and responsibilities

Since both enterprise architects and developers would benefit if it was possible to work more closely together [22], thereby building more trust and understanding of each others' work, we propose a community consisting of both developers and (enterprise) architects who meet on a regular basis. Especially in large-scale agile software development, expertise might be scattered across multiple locations and teams, making networking and knowledge-sharing between teams critical [81]. Šmite et al. summarize various mechanisms that can support the development of knowledge networks in large-scale agile development, e. g. collaborating closely with experts outside an individual team, facilitating communities of practice and providing suitable communication infrastructure [107]. Our approach fits well with these proposed mechanisms. The goal is that this community creates, validates and decides on architecture principles as well as handling changes to existing principles and guidelines. From our point of view, it makes sense to not restricting the community to focus solely on architecture principles and guidelines. On the contrary, it is probably suitable in most cases to embed the topic of principles and guidelines into a broader agenda. Such an agenda could be designed to enable teams exchanging information on architectural topics and presenting their architectures, status and findings, thereby receiving validation and feedback from other teams and identifying needs for optimization. In case the participants notice a broader relevance of these discussions or identify drivers for new potential principles or guidelines or changes to existing ones, they can act accordingly.

To keep the community relatively small in size, so that the decision-making ability is not lost, it can be useful to have one team member from each team joining the community, as a representative of that team. Those representatives do not always have to be the same person and they do not necessarily have to be in a lead role since the community could also benefit from different opinions and viewpoints. The same applies to the architects who are joining the community. Teams should have the right to elect the representative freely. In some specific cases, it could be necessary to request a certain team member who has valuable knowledge or insights on certain topics that

are on the agenda for a meeting. Nevertheless, in general, the community should be open for everyone who is interested and not enforce participation.

As described previously when discussing involved stakeholders, in terms of responsibilities, the goal is that the architects are facilitating the community, but the community is making the decisions. Or as Sam Newman puts it in his similar approach on creating a group of architects and developers, working closely together: "The architect is responsible that the group works, but the group as a whole is responsible for governance" [82]. This also comes with a necessity for a shift in the mentality of the architect. In their concept for a more adaptive enterprise architecture management by Korhonen et al., where the enterprise architect is also seen in a more collaborative role, the architect has to build humility and respect for the people that he is helping and also believe in their capacity to grow and learn [58]. Regarding the question of what areas the community is responsible for, the idea of the community is not that every aspect must be discussed in detail and that teams must necessarily participate in all areas. Rather, due to time and capacity constraints of the participants, there could be the decision made that teams do not want to deal with certain topics and would like to leave them in the responsibility of the enterprise architects or other stakeholders. And on the other side, the decision could be made that certain topics are left for the individual teams to decide, without the need for cross-team principles or guidelines. Nevertheless, something of value was achieved: instead of leaving certain areas almost randomly to "emergent architecture", there has been a conscious and well-founded decision made on which areas are left to emergent architecture and which areas are left to intentional architecture. This also matches one of the advantages of architecture principles stressed by Greefhorst and Proper, who state that "By focusing on the essential requirements, the use of architecture principles allows/invites enterprises to think carefully about what to regulate in a design-first style and what to leave up to emergence, or to even take measures that enable desirable emergence." [37] In general, the community should be worried about what happens between the teams and be liberal about what happens within an agile team.

A suitable frequency of community meetings is strongly dependant on the specific organization and can be suggested for the first time by enterprise architects and can later be adjusted according to the community needs.

As mentioned earlier, we are aware of the fact that especially in larger organizations, enterprise architects might not be the originators of all principles and guidelines that are affecting development teams. There might be other governance entities outside of the enterprise architecture department, or groups specifically dealing with certain areas, e. g. (information) security, data protection and privacy or legal requirements. Those would have to be taken in account as well and included in the collaboration, when necessary. Since the above mentioned examples mostly build their work on

legal requirements, there is less need for discussion and close collaboration within the community, but it has to be ensured that the information flows properly between those departments and the community of architects and developers, which again should be facilitated by the architects. In the end, it is important that a "single source of truth" is built. That means that for every guideline that a stakeholder group would like to set for agile teams, they would have to go through the community. This ensures that guidelines are not created over the head of agile teams and that agile teams can rely on a single point of contact regarding guidelines, saving a lot of time and effort.

5.2.3. Collaborative process steps

The following process of our approach is based on the established process of Greefhorst and Proper [38], as summarized and presented in Section 2.3.4. Where in the original process, the life cycle of principles is mainly driven by enterprise architects, now all stakeholders affected by guidelines are continuously involved, so in our case of large-scale development, we closely involve the agile development teams. Nevertheless, even if Greefhorst and Proper do not explicitly focus a bottom-up perspective and collaborative traits in their approach, they stress the need for the collaborative involvement of stakeholders multiple times in their book, as reflected by the following exemplary quotes:

- "These processes are best performed collaboratively to ensure involvement and commitment of stakeholders." [37]
- "It is of the utmost importance that principles are formulated in a collaborative process involving all key stakeholders. More research is needed into effective ways to organize these collaborative processes." [37]
- "Therefore, more research is needed into ways of better dealing with the top-down versus bottom-up and design-first versus emergence 'game'." [37]

Because of this need, we would like to contribute the following resulting steps for the process being used in a collaborative approach, illustrated in Figure 5.2. The titles of the process steps mostly match the steps presented by Greefhorst and Proper [38], with the difference of our suggestion to merge the "specify principles" and "classify principles" steps and our renaming of the "validate and accept" phase, which we call "vote and accept", to put more emphasis on the community-driven decisions. The detailed descriptions of the steps have been adjusted to take better account of the inclusion of the bottom-up perspective.

Important to note is that the process is not meant as a strict specification of an order of steps that has to be followed in a linear fashion. Rather, **it is meant as a**

recommendation of possible activities that the suggested community should deal with. Those activities can occur in different orders or with different entry points, depending on the situation. For example, some drivers might already exist, so the community would directly move on to the specify principles and guidelines step when dealing with these existing drivers. In other cases, there might already be existing principles or guidelines, although they might lack a proper specification. In this case, the process would start with the third step, which revolves around specifying and classifying the principle or guideline. Additionally, there might be cases where most principles and guidelines are already sufficiently defined and the community focuses mostly on the application of principles and guidelines as well as managing their compliance and changes. Nevertheless, we choose to present the suggested activities of the community in form of a process to achieve a better fit with the established architecture principle process as well as a clear structuring and logical order of activities.

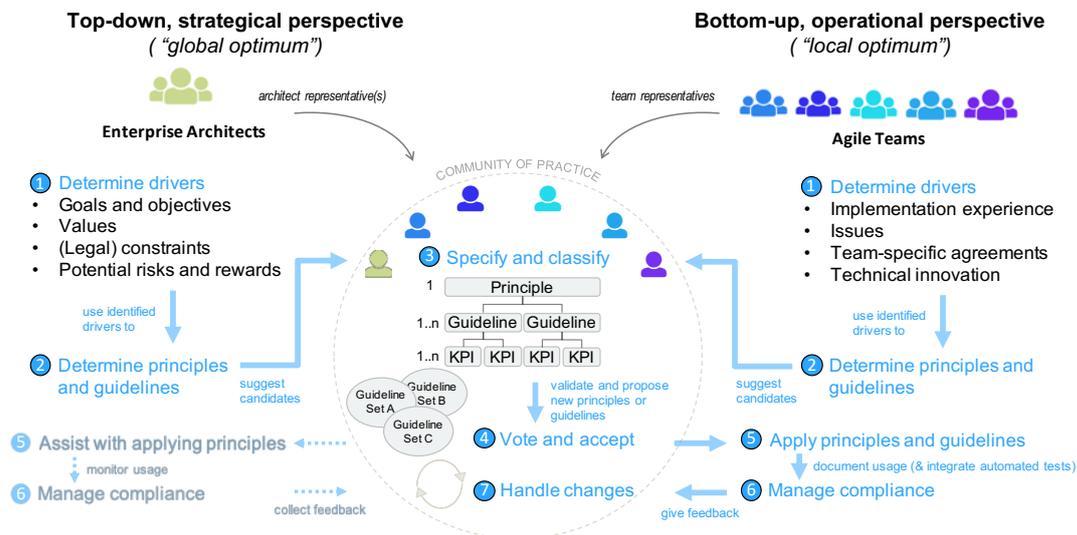


Figure 5.2.: Collaborative approach overview

In the following, we describe the resulting steps in more detail, with a focus on the difference to the original process in Section 2.3.4.

Step 1: Derive drivers

According to Greefhorst and Proper, principles without drivers are pointless. Determining drivers revolves around analyzing relevant sources and collecting suitable input for deriving architecture principles, e. g. goals and objectives, values, issues, risks,

potential rewards and constraints [38]. This often requires certain levels of analysis, e. g. conducting interviews or organizing workshops [38]. The key difference within this process step in our approach is that not the enterprise architects alone are responsible for identifying those drivers, keeping them up to date and ensuring their quality, but through the community, enterprise architects and agile teams are responsible together. In our view, this has some significant advantages. Most importantly, it opens up new types of drivers and ensures easier access to some drivers. Agile teams can contribute their implementation experience as a basis to derive principles and guidelines. This bottom-up perspective can provide valuable input to principles and guidelines by taking in account the practical experience of the teams. They experience first hand what implications the governance activities actually have and how they ultimately affect actual implementation. This provides valuable insight into which governance intentions work and which ones do not work. In addition, already existing, team-specific internal agreements or agreements from previous projects can lead to architecture principles and guidelines. Those team-internal agreements might not always be explicitly documented, but are often a result of the selection and agreement of a suitable solution to a reoccurring challenge that a team has faced over a period of time. This information could be valuable for other teams as well and therefore lead to principles or guidelines later in the process. Furthermore, agile teams are often closer at the heartbeat of technological innovation because of their day-to-day technical work and keeping themselves up to date with the latest developments. This can be valuable input for new or changing principles and guidelines. Furthermore, concrete, tangible issues mostly arise during the actual implementation. Because of their proximity to actual implementation, it is much easier for agile teams to identify and analyze those issues in more detail. This is why we place issues in Figure 5.2 on the right side of our approach, in the main visibility and responsibility of the agile teams. This matches with what Greefhorst and Proper call "operational issues". They state that "Including them as drivers enables operational employees to provide relevant input to the architecture, and thereby involve them in the process" [37]. Involving agile teams so closely in the process as suggested in this approach brings the advantage of shining more light on those issues.

Of course issues will also still arise on a higher, more strategic level, but it will save architects time by not having to do a lot of digging to identify their causes, because the origin of the issue might be on an operational level. At the strategic level, you can observe the implications or effects of issues, e. g. higher costs, delays or low re-use of standardized components. Consequently, enterprise architects can focus on more strategic types of drivers where their expertise and positioning gives them a crucial advantage. These drivers are especially organization-wide goals and objectives, values defined as part of the (IT-)strategy, potential risks and rewards and legal constraints. Guidelines based on legal requirements also often originate

from team-external sources, because teams rarely have the time, oversight and/or qualification to take legal requirements into account. Since legal constraints play an important role in directing organizational decisions, because failure to comply with regulations and laws might lead to fees, penalties or reputation risk [99], these drivers have to be documented with additional care. Later in the process, it will be important to distinguish between guidelines that are based on legal constraints and therefore required to fulfill a law or regulation, because those are types of guidelines that might have to be excluded from a community vote in the fourth step of the process. The positioning of enterprise architects with their access to multiple teams also gives them the advantage to identify cross-team patterns more easily. Due to the collaborative nature of the proposed community and process, the increased interaction between teams should also help those teams to be able to identify such recurring patterns.

Step 2: Determine principles and guidelines

As described earlier in Chapter 2.3.4, the identified drivers can be used to derive principles, which encompasses three key steps: Generate a list of possible principles, select the relevant principles from this resulting candidate list and then use the chosen candidates to formulate the actual principle statements [38].

These key activities also remain the same within this step in our collaborative approach. The only key differences now in our approach are that agile teams are directly involved in the question what principles are relevant or not and that not only principles can be derived from drivers, but also already more specific guidelines. Furthermore, through the newly introduced drivers described in the previous step, these additional drivers now have also to be taken in account within the principle and guideline determination. Since the drivers that are identified by agile teams are usually closer to actual implementation, as described before, these drivers could also directly result in guidelines. This is especially conceivable when teams have identified more specific issues that can be directly translated into a principle that guides actual implementation, therefore into the kind of principle what we classify as a guideline. In this case, it might be worth analyzing if there is already an existing principle that fits the resulting guideline as a more generic, overarching statement or if there is a need for creating a more generic principle as well. This relationship between principles and guidelines is discussed in the next step of the process.

When selecting relevant principles of the list of potential candidates, we believe that principles and guidelines that focus on the interplay between various teams or services are the ones with higher importance and higher binding nature ("has to be done"). Guidelines that are providing solutions to a specific implementation usually should be "may be done", because those are best-practices that originate from teams. In these

cases, those guidelines could help other teams by providing them with ways to solve their challenges, but it should not limit their ability to choose a solution themselves. One simple test that could be used to support the decision on the binding nature of a guideline is to look at the consequences of not fulfilling guidelines: In the first case, the direct consequences affect only the team itself, which indicates a lower priority and binding nature. In the other case, it affects not only the team, but other stakeholders (e. g. because the team decides on a different logging and monitoring stack than the standard, they now cause additional work for other stakeholders to incorporate them in the existing, overall monitoring). We suggest on relying on the team to make the right local decisions. But everything that affects more than the local team space is highly relevant for the community. As mentioned earlier, the community should be primarily worried about what happens between teams and be liberal about what happens within a team.

Step 3: Specify and classify

In contrast to the generic approach by Greefhorst and Proper [38], we combine the two phases of "specifying principles" and "classifying principles". We merge the two steps not only to reduce the total number of steps in the approach, but also because we would like to stress the need to consider the classification of the principle during the specification, e. g. to determine the target groups and align the specification of the principle or guideline accordingly. This means that a principle or guideline is specified in a way that the resulting specification is most useful for the intended target group. In our approach, this is of higher importance because we put more emphasis on the specificity of principles. The importance of this specificity is also recognized by Greefhorst and Proper who state that when enterprises would like to use principles to actually limit design freedom, they need to be specific enough and formulated in a way that allows assessing the compliance with those principles [37]. We would like to ensure this specificity in our approach by differentiating between principles and guidelines. Furthermore, we propose defining fulfillment criteria for guidelines in the form of KPIs, meaning measurable values that are decisive for compliance with the guideline. This sets an important basis for a possible automated testing of guideline fulfillment in the future. As illustrated in Figure 5.2, we propose to transform a principle into one or multiple more specific guidelines, if necessary. This necessity arises when one of the following two criteria are met. Either, the principle is too broad or unspecific to guide actual implementation and as a result, stakeholders feel confused or do not feel like the principle gives them concrete value. Or the principle is so broad or abstract that it is difficult to define actual fulfillment criteria, let alone concrete measurements that could indicate the fulfillment level of a principle.

These guidelines typically then describe different ways on how a principle can be implemented. Furthermore, if possible, one or more KPIs should be defined for each guideline. There will be principles that will be thought to specify into more specific guidelines and there will be guidelines whose compliance criteria cannot be easily expressed as measurable values. In these cases, it is not worthwhile to enforce a specification into guidelines or KPIs. But the community should critically question if the principle or guideline can provide practical value and relevance if it cannot be further specified or measured. Nevertheless, it is plausible that there are principles or guidelines that do not need to be further specified and still provide added value. Because depending on architecture maturity and organizational culture, it may be sufficient in certain situations to forego specifying the principle and just use it to achieve a common understanding and commitment for certain issues, instead of actually restricting design freedom [37].

We consider this relationship as very interesting because if maintained correctly, valuable observations can be made. In this way, it is much easier to understand how concrete strategic goals (as part of the drivers and principles) affect actual implementation, which concrete measures (guidelines) are taken to implement these strategic goals and how well these measures (KPIs) are adhered to. It could be very insightful for the management to be able to visualize such a relationship, to get a better understanding of the concrete decisions that strategic goals lead to in lower, operative levels. Thereby, management could trace how more generic objectives and values or "meta-principles" (e. g. standardization, modularity, reusability, interoperability, integration, simplicity, compliance, data consistency [40]) are actually implemented on lower levels. Through the use of classifying principles and guidelines, these statistics and relationships could also be analyzed based on certain categories or target groups. On the other side, a potential limitation of this aspect is that the relationship between principles, guidelines, and KPIs is only rarely strictly hierarchical. In most cases of existing principles and guidelines, there are dependencies between multiple principles and guidelines, as well as the possibility of KPIs that can measure the fulfillment of parts of multiple guidelines. The proposed mapping would also involve a significant time and coordination investment, therefore further research would have to weigh up costs and benefits.

For classification purposes, we suggest to define a set of target stakeholder groups and then classify principles and guidelines based on the target groups. Of course, a principle or guideline can be relevant to multiple stakeholders. But in only rare cases are they relevant for all possible target groups. This causes additional complexity and wasted effort. If stakeholders are confronted with a variety of content that consumes time but is not relevant to them, it could also reduce the overall acceptance of principles or guidelines. In addition, it makes it more difficult to clearly identify the principles and guidelines that are actually relevant. In general, due to their more abstract character,

principles are more relevant to a larger target group. Guidelines, on the other hand, tend to be more limited in their relevance because of their higher degree of specificity. To guide actual implementation, there is more need to take into account the type of implementation that should be targeted.

The specification and classification could be done in a small group or by an individual participant of the community, but the results should be validated within the community. A form of validation will also happen in the next step, during the vote and accept phase, although it would be preferable if principles and guidelines already fulfill a certain quality level before they are put to the vote.

Step 4: Vote and accept

Because of their high importance, a high quality of principles and guidelines has to be ensured. Therefore principles and guidelines need to be validated before they get accepted into a pool of principles and guidelines that are used productively. Important to note here is that all the previous steps both should include some sort of validation. Greefhorst and Proper point out that the validating process itself can be highly standardized and include specific roles. As an example, they mention discussing and agreeing on the principles in an architectural board with management representatives of all major departments [37]. In our approach, we believe that the community should be responsible for making the final decision on accepting a principle or guideline. This means that there has to be a discussion and vote whether it can be accepted, needs further refinement or maybe even has to be declined because there are solid reasons which speak against the principle or guideline. The result of these votes should be documented for further reference. For the enterprise architect, this will likely mean an increased required effort and commitment to present proposals convincingly. Without explaining the rationale behind the suggested principle or guideline, it might be difficult to get the support of the community. We suggest that the effort is worth it, because as observed during the case study, principle and guidelines that do not have the acceptance of the target stakeholder are likely to be ignored or to be bypassed anyway. Even if the proposed principle or guideline does not provide direct value to the individual agile teams, with the awareness of the reasoning and the benefits at a different part of the organization, the community can be convinced of the introduction anyway, presupposed the community already has a certain maturity level. Depending on this maturity level and the characteristics of the organization, the enterprise architects might provide strong, detailed governance or gentle directional guidance. The current state of the level of authoritarian control or guidance has to be taken into account during the establishment of such a community. A democratic vote, meaning that each representative of the agile teams has the same voting power as each enterprise

architect within the community, requires a certain level of organizational maturity. In organizations that have only recently begun to transform into a more agile environment, it might be required to steer such a community more strongly, e. g. by veto rights of the enterprise architects or power to overrule the community. To go back to the metaphor of the town planner, who more accurately represents the role of an IT-architect, as described earlier, he should only steer the broad direction and not control the details. But "if someone decides to build a sewage plant in a residential area" as Sam Newman describes it, "he needs to be able to shut it down" [82]. This means that, in extreme cases, there might be the need for someone within the community to intervene, but it should be the clear exception to the rule. The more mature the community, the less likely such situations should occur. Some organizations might benefit from not taking such measures to overrule the community at all, leaving full power to the community and therefore ensuring its' credibility and commitment.

As explained earlier, not all proposals can be run through such a vote. If the main driver of a principle or guideline is a legal requirement, a vote whether such a principle or guideline is necessary does not make sense and therefore can be omitted.

Step 5: Apply principles and guidelines

In research for their process, Greefhorst and Proper conclude that there is surprisingly little guidance on how stakeholders actually use architecture principles in the creation of their own artifacts [37]. We believe that the specification of principles to guidelines and KPIs should make the application much easier. At least, it should make it much clearer what a more generic principle actually means for implementation. A crucial advantage is also that the people who are executing the work that is governed by principles and guidelines are actually the main stakeholder in our approach. Naturally, this should improve the applicability of those principles and guidelines. It could be valuable to document the solution to specific guidelines, as it could serve as an example for other teams and provide insight into the solution requirements that implement those guidelines. Applying a guideline should be as easy as possible for the teams and in the best case, it should be even easier than not applying the guideline. This could be achieved by providing reusable templates, components, guides, code examples etc.

Step 6: Manage compliance

As in step 4 "vote and accept", the specific degree on how much self-responsibility the community can have in this step is heavily dependant on the organization. Depending on the maturity level of the organization regarding agile processes and mindset, it is most cases probably indispensable to support teams to achieve self-governance,

especially in the beginning. A way how this can be achieved is to show teams ways to get a clear overview of the guidelines they have to fulfill and let them track their current compliance status on these guidelines. We decided to use a web application for this purposes, as described later in Chapter 6.

The advantage of our approach is that through the involvement of stakeholders who execute the work that is being governed from the beginning, there is a higher chance that self-governance will be sufficient. Those stakeholders are aware of principles and guidelines because they coined them. Giving teams the responsibility to follow guidelines and therefore also the freedom to decide against using a specific guideline can also improve acceptance. However, we see the need for documenting these decisions. The community needs to know when their plan is not being followed. A very low compliance degree of a guideline across all teams can be an indicator that there are existing issues with a guideline, either in implementing the guideline or in the specification of the guideline itself. Documenting exceptions can be very valuable and is both mentioned by Greefhorst and Proper [38] as well as Newman. As he states it, "tracking exceptions may be vital to ensure that the rules put in place properly reflect the challenges people are facing" [82]. If enough exceptions are found, it may make sense to change the principle or guideline. Therefore, enterprise architects could have an eye on whether those decisions are being documented, without strictly monitoring every guideline and a team's compliance with that guideline. Time-consuming, extensive architecture reviews as in traditional governance processes are not necessary anymore.

In the best case, only guidelines remain in our approach that are inherently useful for the affected target groups and complying with a guideline is also the easiest thing to do. Having to manually and continuously check the compliance to all relevant guidelines can be a significant effort. One way to reduce this effort is by automated testing, preferably directly integrated into the build pipeline. The goal would be to automatically test some of the guidelines. Amongst other things, this requires measurability of guidelines, which we would like to achieve by specifying KPIs. The web application presented in Chapter 6 aims to build a basis for automated testing of guidelines and could be extended to integrate various data sources for automated compliance testing of certain guidelines.

Step 7: Handle changes

It is crucial to collect feedback from the involved stakeholders to get a more genuine impression on the actual consequences of governance activities [15]. Furthermore, it is important to actively involve all stakeholders during the feedback phase to be able to profit from the diversity of knowledge in an organization, which can be a valuable input for architectural decisions [15]. In general, small changes can be applied by

directly altering a principle or guideline, bigger changes, on the other hand, might require a new principle or guideline. The advantage of principles and guidelines is that they are mostly self-contained, which provides the opportunity for small-scale, iterative releases. Greefhorst and Proper propose a feedback mechanism where people can comment on architecture principles, request changes or discuss with peers on specific experiences [38]. Our proposed community provides the opportunity to do so. The key difference is that, again, not only the architects are responsible for monitoring causes for changes and taking care of these changes, but the community is. Since feedback that requires the change of a certain principle or guideline will naturally often arise during applying that principle or guideline, therefore during implementation, it is a significant advantage to have the agile teams so closely involved in the approach. In their practical approach, Greefhorst and Proper state that "It can even be valid to adjust the architecture principles based on insight originating from specific situations" [38], which fits well with including the experience of individual agile teams in specific situations in the change process.

5.3. Addressed challenges and solution requirements

In this section, we summarize the relevant challenges identified in the case study presented in the previous chapter and explain what requirements and solution objectives result therefrom. This ensures that we create solution artifacts that provide actual value by tackling multiple existing challenges. We describe solution requirements that we consider both in our collaborative approach presented in this chapter as well as in the web application, presented in Chapter 6, that supports the approach. The approach does not necessarily have to be supported by a web application, as we describe in Section 6.1. Therefore, it would be possible to implement certain requirements in a different way, for example with the help of physical Kanban boards, whiteboards or sticky notes, or other digital collaborative tools, e. g. a community and wiki platform. Nevertheless, our suggestion of a software-based support comes with certain advantages, such as scalability, maintainability and expandability with automated tests. These advantages and the reasons for our decision to support the approach with a web application are also described in more detail in Section 6.1.

No single source of truth (C1)

As observed in the case study, a key challenge is that there is no single source of truth for principles and guidelines for affected stakeholders. Principles and guidelines are often spread throughout multiple departments and documented in different ways and locations. Therefore, one of the main requirements of a solution is to create a

"single source of truth" that unites all efforts regarding architecture principles and guidelines that are relevant for agile teams. It should also provide a central place for managing and documents principles and guidelines as well as related information. This allows for quicker decision making and feedback cycles as well as less effort and time consumption for identifying existing guidelines and relevant contact persons.

In our approach, we address this by proposing a community that has the main responsibility for all principles and guidelines that are relevant to agile teams. Furthermore, we introduce a web application as a common repository for documenting guidelines themselves as well as their fulfillment status and other related information, e. g. feedback on those guidelines.

Multiple involved stakeholder groups (C2)

A large amount of potential stakeholders involved make communication more difficult, requires additional time and effort for teams to identify suitable contact persons and thereby further aggravates the additional time and effort problem (C8). One of the common problems in this area is also that teams have to await architectural decisions before they can continue with certain tasks. There is also the risk that efforts regarding principles and guidelines are not harmonized and that different stakeholder groups have different goals and priorities in mind, possibly causing conflicting and interdependent principles and guidelines that could further delay and obstruct agile teams in their daily work. A key requirement derived from this challenges is the need to involve all relevant stakeholders, so that governance efforts can be harmonized and integrated together. This enables potential conflicts of interest to be identified and resolved at an early stage before they lead to problems during implementation.

The resulting solution objective is to invite all key stakeholders to the community on a regular basis, and the stakeholders more specialized to certain areas (e. g. security, operations, etc.) when necessary. In addition, everyone interested should be able to access the tool support for the related documentation.

Lack of awareness and compliance with guidelines (C3)

The challenge describes the problem that guidelines are often not applied in the actual implementation, either because of a lack of awareness of certain guidelines or because the target stakeholders deliberately ignore them. This could be caused by the lack of relevance or applicability of a guideline (C6) or the additional time and effort required to identify, understand and apply existing principles and guidelines (C8).

Consequently, it is required to improve awareness and ensure the compliance with principles and guidelines. A crucial question is how exactly architecture compliance of

implementation projects should be ensured [42]. Since enterprise architecture management might benefit from a tighter integration into agile methods and processes [42], we propose a collaborative approach with tool-support as presented in this thesis.

Low acceptance of principles and guidelines by agile teams (C4)

As described in the previous section, principles and guidelines are sometimes getting ignored. This can be related to the issue that agile teams usually prefer independence and self-responsible decisions, and do not like to be controlled or governed. In general, top-down driven management initiatives have a lower acceptance in agile environments [29].

Therefore, one of the central requirements is to not solely rely on a top-down, authority-based governance and control approach that gives the full responsibility of creating and managing guidelines to a single person or small of group of people, mainly consisting of management stakeholders or lead architects. Instead, all relevant stakeholders should be included in the governance efforts, thereby extending the top-down approach by a bottom-up perspective.

In our approach, we intend to solve the acceptance problem by giving agile teams a say in decision making and governance. Teams should be able and actively empowered to define solutions on their own, while still considering certain limitations set by the current and future IT landscape [42].

Unclear implications of guidelines (C5)

Another common challenge is that the implications of principles and guidelines are often unclear, meaning that it is often ambiguous what a principle or guideline really means during actual implementation and how exactly it can be fulfilled. This is especially the case for architecture principles, which are more general than guidelines (as explained in more detail in Section 2.3). Additionally, there is the question of whether the adherence to the principle or guideline is mandatory or more of a recommendation that can also be deviated from.

In our view, architecture principles face this crucial challenge due to two contradictory goals they are intended to fulfill. On the one hand side, they should be applicable to all solutions that match the scope of the architecture [37], which requires a certain level of abstraction and generalization. On the other hand, this more generic nature limits their applicability, because principles have to be specific enough to get applied. Greefhorst and Proper state that "Architecture principles are still fairly generic, which does not position them as strategically and thereby effective as they could be" [37] and "It is important to carefully determine the extent of generalization that is needed. You

should not generalize too much since that can have a counter-productive effect" [38]. As a result, there is a trade-off between generalizability and specificity of architecture principles. Architecture principles should be specific enough to get applied, but general enough to apply to all solutions that match the scope of the architecture.

This is the reason why we introduce principles and guidelines as separate terms in our solution. Since the two goals of generalizability and specificity are clearly contradicting each other, it is difficult to reach them at the same time. We believe that the scope and target group of architecture principles play an important role in this matter. Because to find the sweet spot in the trade-off between a high generalizability and enough specificity to guide actual implementation, we have to use principles and guidelines scope-related. Meaning there is not one single way of how to fulfill a certain principle, but multiple ways based on project characteristics and requirements, target stakeholders and so on. Therefore, an additional requirement is to manage target stakeholders and categories of principles and guidelines to be able to map them to project characteristics.

The second major aspect of the unclear implications of a guideline is that it might not be clear what guidelines have to be fulfilled, without any or with only very minor deviation possible (e. g. because of legal requirements), and which ones are more a recommendation and can be adjusted to individual requirements of individual products or teams. A resulting requirement is that it should be made clear whether a certain guideline is a strict regulation or rather a broad recommendation.

A possible solution could be an additional attribute describing the guideline: the liability level. The liability level signifies how binding a guideline is. The terminology that we propose is borrowed from the field of requirements engineering, where it is typical to describe the priority of certain requirements levels with terms as "must" or "shall", "should" or "recommended" and "may" or "optional" [17]. In regard to guidelines, a "must" is justified by a regulatory or legal issue that results in the need for a guideline. It could also be justified by an important internal policy, but we advise caution doing so because of the following reason. "Must" guidelines are intended as "non-negotiable" rules which ensure compliance. Their number should be kept to a minimum since they would also be integrated into auditing and revision processes. For the community in the collaborative approach, it might also important to define something like a liability level for specific guidelines early on, so that governance efforts that are based on legal requirements cannot be stopped by a community vote.

Lack of relevance and applicability (C6)

Next to the unclear implications of guidelines, another important challenge is the lack of relevance and applicability of architecture principles and guidelines. This can have

different reasons. One of the common reasons is that architecture principles are often still fairly generic [37], as detailed in the previous challenge. Another possible reason is the distance of guideline creators to the actual implementation. Decisions might be made that directly or indirectly affect implementation details, without detailed knowledge and insight on these areas, thereby negatively impacting the practical applicability. In addition, the comprehensibility might be a reason for a low applicability, since principles can possess a high complexity and low accessibility [40]. But even if the principle is well specified and giving clear guidance for actual implementation (what we refer to as "guideline"), it could be simply not relevant for a certain stakeholder group. A principle or guideline cannot continually be relevant, useful and applicable for every single team. There are always exceptions to the rule due to team or product-specific requirements or special circumstances that might make a guideline irrelevant or sometimes even lead to negatively impacting a team.

To address the low relevance and applicability, a key requirement is to enable operational employees to provide relevant input to the guideline creation. As Greefhorst and Proper [37] state, including operational issues and aspects "provides an opportunity for the architect to contribute to problems that people are confronted with in their daily work, and is an important step in the acceptance of architecture principles." Generally speaking, the requirement is that the groups who eventually have to apply the principles or guidelines should be involved in the creation process to ensure better applicability. This also tackles the challenge of the low acceptance when those stakeholders are not involved in the process, as explained earlier. We implement this requirement by building our approach around the core idea to include the bottom-up perspective over the whole life-cycle of architecture principles and guidelines.

As mentioned in the previous challenge, architecture principles and guidelines have to be specific enough to be relevant for actual implementation. Also, target stakeholders of principles and guidelines should be clearly defined and principles and guidelines should be categorized accordingly. We propose to use tags as an easy solution for categorization and use those tags to map project and team characteristics to those tags, which results in a quick selection of relevant guidelines and an evolving pattern matching that helps to end up with certain guideline sets for different stakeholders.

The problem that not all guidelines are relevant for every team requires the possibility to be able to override existing principles and guidelines and move forward with a team-specific solution. Making architectural decisions is often about trade-offs, and especially in system design, it is not possible to define principles that are always right in all of the cases. This makes also sense from the perspective that the closer collaboration between enterprise architects and agile teams can leverage the knowledge of the experts in the field and leave as many of the decisions as possible to them [42]. Therefore it is important to let teams deviate from the standard. Nevertheless, in those cases

of reasonable deviation, it is of high interest to learn more about the reasons for the deviation. This is valuable because first of all, other teams in the future can learn from the experience and also use similar deviations without redoing the full process. And secondly, if enough deviations are found and it gets clear that multiple teams have similar good reasons for deviating from the principle or guideline, it might make sense to adjust and improve those principles or guidelines. This can result in better versions of those standards and ensures that lessons learned and practical experiences are incorporated into those standards, to tackle and better match the challenges teams face on a daily basis.

The close collaboration proposed in the approach, as well as the feature to document a team-specific guideline rationale and solution in the web application as described in Section 6.3.2, aim to address these problems.

Lack of transparency (C7)

The lack of transparency challenge has multiple facets. One of the main problems is that it is not clear why some principles or guidelines are not being used by teams or if and why they deliberately decided against the usage, as explained in the previous challenge. For future reference, it might be worth capturing these exceptions and fundamental choices of the teams. This problem is also closely related to the lack of awareness and compliance with guidelines (C3). It is not only unclear if a team is using a guideline or not, but also whether they currently fulfill a guideline or not. Another facet of the transparency challenge is that it is not always clear to agile teams why they should adhere to a certain guideline, and what the reasons and arguments are that lead to the decision for creating a principle or guideline. The resulting requirements for a solution are that it has to provide a better overview and documentation on what guidelines are being used by which teams, which guidelines are being fulfilled by which teams and what the rationale behind a guideline is. These requirements are implemented in the tool support by the guideline overview for each team, the guideline statistics page and the ability to document and display rationales for each guideline. Those solution features are described in more detail in Section 6.3. A possible further extension of the tool-support could include the managing of drivers and the mapping to principles and guidelines, which can provide more information on where certain principles and guidelines are coming from, additional to the rationale description. For the time being, the drivers can be easily documented as part of the rationale behind the principle or guideline.

Additional time and effort required (C8)

Enterprise architecture management is a complex field and capacities are usually scarce. Hence, enterprise architects cannot make a vast number of well-informed decisions on a detailed level themselves. Therefore, it is required to shift some of the decision making to the agile teams, which aims to profit from multiple advantages. On one side, enterprise architects have more capacity to focus on areas where they create added value. On the other side, agile teams could become more satisfied and accepting towards governance and enterprise architecture efforts because they have more say in decision making.

But agile teams do not have many resources available as well and should focus on implementing their product or project requirements. Therefore, it is required that the solution provides high usability and as much automation as possible. Of course, a community as presented in the collaborative approach and all its related activities, e. g. creating principles and guidelines, checking their compliance, handling change and maintaining them, etc. consumes time and requires commitment from the involved stakeholders as well. Greefhorst and Proper also note this regarding their generic process for architecture principles, stating that "in practice it remains difficult to apply the process with enough depth and formality, due to limited time that is typically available" [37]. We keep this in mind by trying to extract and focus on the most important aspects of the process. In addition, our collaborative approach comes with the advantage of splitting the workload within the involved stakeholder groups within the community, which should further alleviate the problem.

Some of the solutions presented in the tool-supported collaborative approach already aim to address these challenges. For example, the mapping of project or team characteristics to guideline categories allows for a quicker identification of relevant guidelines. The statistics page automatically aggregates and visualizes the existing information on compliance with guidelines over multiple teams. A web application also provides the basis for automatically testing the compliance with guidelines in the future, because it can access and integrate various data sources. In addition, the community aims to save time in the long run by bringing all related stakeholders together. This solves the often necessary long search and delay in identifying the appropriate contact person for certain principles and guidelines, due to multiple stakeholder groups (C2). Furthermore, this should prevent teams from having to await architectural decisions.

Architectural integration into the platform (C9)

Based on the findings from the case study, it is important in large-scale agile software development to integrate architecture directly into the development platforms, e. g.

by using automated tests of architecture principles and guidelines, providing pre-configured tooling, suitable code templates and scaffolding or the ability to generate the necessary CI/CD pipeline and other important aspects of the development. This is important for making the desirable behavior as easy as possible for affected stakeholders, resulting in a higher compliance to the intended behavior.

Even though there is more research needed on the details on how such an integration can look like, we provide a first step by deciding for the use of a web application to support the collaborative approach. A web application provides the basis for connecting to other data sources that can be used for automated testing of architecture principles and guidelines. Furthermore, the application could be integrated into existing CI/CD pipelines, e. g. to act as a quality gate and check for the fulfillment of certain guidelines before a deployment to production is possible, as described in Section 6.7 about the potential next steps regarding the implementation of the tool support. In addition, since the automated testing of generic concepts would be difficult, we suggest a specification of abstract principles to more specific guidelines and even KPIs, for the purpose of reaching measurable values that can then be used during the automated testing.

Collaboration, feedback cycles and consideration of bottom-up perspectives (C10)

The final important challenge presented in this thesis is the need for a closer collaboration and feedback cycles as well as the consideration of bottom-up perspectives in governance and enterprise architecture processes. Greefhorst and Proper stress that principles can be used as a control mechanism, but should not be mistaken to be solely part of a top-down steering approach [37]. By observing factors that lead to the violation of existing principles and the emergence of the need for new principles, architecture principles "can be used as an indicator mechanism as well" [37]. Therefore the key requirement for the solution is to facilitate a higher degree of collaboration and to take into account the bottom-up, operational perspective. We take this into account in our approach by making the bottom-up perspective of agile teams a crucial part of the overall approach and process. As already mentioned in a previously described challenge, speaking with operational employees and taking into account the bottom-up perspective is important because it provides an opportunity to contribute to problems that the teams are encountering in their daily work [37]. A tighter integration and collaboration also helps to bridge the gap between enterprise architects and agile teams [42] and thereby facilitates a better mutual understanding and acceptance between each other [64, 42]. This also leads to a better ability to rapidly respond to changing needs [50]. The close collaboration between enterprise architects and agile teams can also "leverage the knowledge of the experts in the field and leave as many of

the decisions as possible to them" [42].

Collaboration in large-scale agile development is not only important between enterprise architects and agile teams, but also between different agile teams. A common problem in large-scale agile development are deficiencies in inter-team coordination and communication [117] as well as the need to have an effective knowledge network and collaborate closely with experts outside the team [81, 107]. Therefore, the solution artifacts should also fulfill the requirement to encourage and facilitate better communication between teams. By including representatives of multiple teams in the community within the approach as well as documenting and presenting teams who can help with implementing certain guidelines in the tool support, we aim to fulfill this requirement.

6. Implementation

This chapter describes the prototypical implementation of a web application aiming to support the collaborative establishment of principles and guidelines as explained in the previous chapter. Its features are designed to solve the observed problems described in the previous chapter and to fulfill the resulting requirements in a prototypical implementation.

Due to the time constraints of the thesis, the tool support does not yet support all phases of the presented collaborative approach, but focuses on specific aspects. The focus is to support applying and managing the compliance of principles and guidelines, and to provide a better overview and mechanisms to facilitate more awareness and transparency in regards to guidelines. This prioritization of features was mainly done in respect of the main challenges that the case study organization faces.

In the beginning of this chapter, Section 6.1 explains the motivation and rationale for developing a web application. Subsequently, Section 6.2 briefly outlines the technical requirements and technology selection. The main views and core features are presented in Section 6.3. Afterwards, in Section 6.4, we explain how social design principles influence the design of those features and how they could be used in the future. Section 6.5 and Section 6.6 present a brief overview of the system architecture and a class diagram of the application.

Finally, Section 6.7 contributes ideas for potential extensions and further development.

6.1. Motivation for a web application

Of course, the development and usage of a web application is only one possible answer to the described challenges. To tackle the described challenges and to achieve the solution goals and improvements, other ways to support the collaboration between agile teams and enterprise architects regarding architecture principles and guidelines could be used as well.

The decision for a web application during this thesis is a conscious choice because of the following arguments that in our opinion outweigh the positive aspects of other possible supportive tools (e. g. Kanban boards, wikis, issue trackers or other project management tools):

6. Implementation

- The possibility of a web application to communicate with other tools, development platforms and databases for data collection, for example to provide the basis for running automated tests for certain guidelines
- Other forms of automating or simplifying processes, as described in more detail in the next paragraph
- The possibility to integrate the web application into existing build pipelines to use the fulfillment of various guidelines as a quality gate for teams to deploy on production
- The scalability, because a web application can better support larger communities and provide means to have votings, feedback cycles and other collaboration across multiple teams
- Accessibility and portability, the possibility to access the documentation and information on principles and guidelines from any location and from different devices
- The interactivity and possible implementation and usage of gamification aspects and social design principles to encourage contribution and participation and other desirable behavior

We believe that well-maintained wiki pages could be sufficient in some cases for supporting the collaborative approach, documenting fundamental choices and making them easily accessible. But especially features that revolve around automation can provide a significant advantage over a wiki. Some examples are listed in the following:

- automated testing of guideline fulfillment
- automated quality gate check, such as a HTTP call from a build pipeline to the application when a specific team likes to build and deploy their application to production, to verify if mandatory guidelines for deployment are fulfilled
- fast creation of new teams, automatic mapping of guidelines to that team based on project characteristics and tags
- automatic generation of statistics
- automatic generation of reports for auditing reasons
- recommendations on new guidelines based on project characteristics

But no matter which kind of tool support is chosen, to establish the tool-support in an organization in the long term, it has to be embedded into a broader process and scope, e. g. by being part of a community of practice similar to the one presented in the previous chapter, or part of sprint retrospectives, architectural spikes or should be taken into account during the evaluation of teams by the management. In line with the first agile value, the individuals and their interactions come first[14], and the process and tools should support that collaboration and communication.

6.2. Technical requirements and technology selection

The technical requirements for the prototype are straightforward. An important requirement for the case study organization is that the application can be easily deployed to the organization's cloud infrastructure, which is based on Pivotal Cloud Foundry¹. Ideally, it should use the standard technologies that are suggested in the organization, which are the Angular² framework for the front-end and Spring Boot³ for the back-end server. To comply with these guidelines, we decided to use these technologies in their latest versions (Angular v6, which was upgraded to Angular v7 during the time frame of this thesis, as well as Spring Boot v2.1). This enables us to use the standard Cloud Foundry buildpacks that are provided by the case study company. As a build tool, we use Jenkins⁴ to be able to build and deploy the two applications effortlessly to the cloud infrastructure and thereby enabling continuous deployment. Furthermore, we use a MySQL⁵ instance which is provided by the organization and which we bind to the backend application in Cloud Foundry, automatically delivering the service instance credentials to our application in an environment variable. As the UI component library, we use the latest Angular Material⁶. It includes material design components specifically for Angular. The organization guidelines stipulate that the internal UI library should be used for all design aspects, but due to the goal to open source the application at some point and the internal library is closed source, Angular Material is chosen because it is open source. For responsiveness across devices, we base our layouts on Angular Flex-Layout⁷ which provides a responsive layout API for Angular applications using CSS⁸ flexbox and Media Queries.

¹ <https://www.cloudfoundry.org/>

² <https://angular.io/>

³ <http://spring.io/projects/spring-boot/>

⁴ <https://jenkins.io/>

⁵ <https://www.mysql.com/>

⁶ <https://material.angular.io/>

⁷ <https://github.com/angular/flex-layout>

⁸ <https://www.w3.org/Style/CSS/Overview.en.html>

6.3. Main views and core features

This section introduces the core features of our prototype, based on the main views in which they are used within the application. The features were carefully designed and implemented based on the challenges identified in Section 4.2 and continuously discussed, adjusted and extended over the course of the case study and prototype implementation, as described in our research methodology in Section 1.3. In addition to the core features presented in the next subsections, the application includes further miscellaneous features, e. g. user and team management, authentication and authorization, settings for guidelines, tags and belts as well as other supportive features.

6.3.1. Overview of the guidelines of a specific team

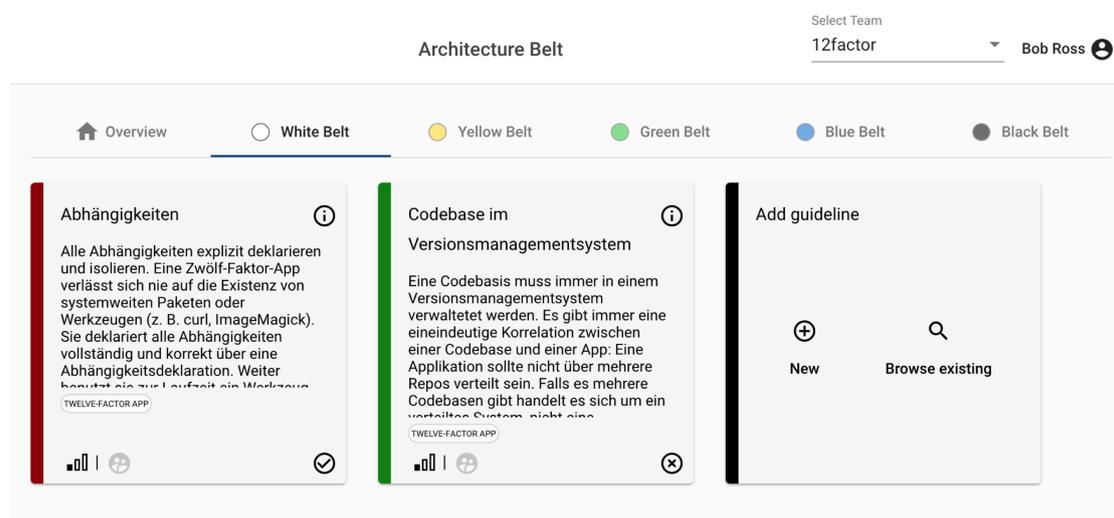


Figure 6.1.: Team-specific guideline overview with belt category selection and fulfillment status of each guideline in the selected belt

The core feature of the application is the guideline overview for each team. It enables the user to get a quick overview of the guidelines that are relevant to his team and the fulfillment status of these guidelines. The team-specific guideline overview screen can be seen in Figure 6.1. The most apparent trait are the tabs on the top which show differently colored belts. This is used as a categorization for the guidelines and can be used to switch between displaying guidelines that are assigned to the respective belts. The term "belt" is a metaphorical reference to the different ranks in martial arts and is part of our gamification approach. In the same manner a martial artist can achieve belt

after belt, starting from the white beginner belt until the black champion belt, teams can rank up their belt as well. Instead of having dedicated tests to rank up as in most martial arts, teams can rank up by fulfilling guidelines.

Each tile on the screen represents a single guideline which is assigned to the currently selected team of the logged in user. The colorized border on the left of each guideline tile represents the current fulfillment status of the respective guideline. A green border indicates that the currently selected team has set the guideline status to "fulfilled", a red border indicates that the team is not fulfilling the guideline yet. The tiles on this overview screen aggregate important information on each guideline. On the top, they show the title of a guideline, followed by a description. To achieve an appropriate overall size of the guideline tile, not the full-length description is displayed, but only the first few lines. This space constraint ensures that enough guideline tiles fit on the screen, even on smaller devices, to achieve a helpful and quick overview. The tags that are assigned to a guideline are displayed right below the description. We consider the tags as crucial information on this overview page because it gives a quick impression what category the guideline belongs to. Below the tags are icons of teams that already fulfill the guideline. This can be valuable information for teams who are looking for help on how to implement certain guidelines in their project or product. With this information, they can approach other teams and ask for support in a more targeted manner. The team icons are currently only indistinguishable placeholder icons, but a tool-tip message reveals the team name when hovering over the icon. In the future, teams could use their team logos or a team icon could be generated from the initials of the team name. The expert teams are also displayed in the guideline detail screen. The information icon button on the top right corner leads to this guideline detail screen. Previously, the whole guideline tile was linked to the guideline detail screen, but this turned out to be less intuitive for the users and caused a lot of accidental clicks. At the bottom right corner, the guideline tile shows an icon button to change the fulfillment status of the guideline. According to the current guidelines status, the button is either used to set the guideline status to fulfilled or not fulfilled. A check-mark icon and cross-mark icon symbolize the two different actions.

6. Implementation

← Customer Journey fulfillment

Please enter the links to the required artifacts to fulfill this guideline:

Artefact name	Artefact link
Customer Journey	🔗

Confirm fulfillment Close

Figure 6.2.: Providing a link to fulfill the type of guideline that requires an artifact delivery

When trying to fulfill a certain type of guideline that requires an artifact, the user is prompted to provide the link to the created artifact which is related to the guideline fulfillment, as presented in Figure 6.2.

Architecture Belt

← Add guideline

Enter a title *

Enter a description *

The description is the visible text on a guideline card.

Select the severity

SHOULD

Select a belt

White Belt

Yellow Belt

Green Belt

Blue Belt

Black Belt

Rationale and Solution

Tag selection

Add Cancel

Figure 6.3.: Dialog for adding a new guideline

For each belt, an additional "add guideline" tile is displayed that can be used to add a completely new guidelines or add existing guidelines from the general guideline pool. When choosing to add a new guideline, a new dialog is opened for creating a

guideline, as it can be seen in Figure 6.3. When selection to add a new guideline from a certain belt, the belt of the new guideline is preselected with this belt, but it can still be adjusted in case the new guideline should be added to a different belt than the currently selected one.

6.3.2. Detail screen of the guideline of a specific team

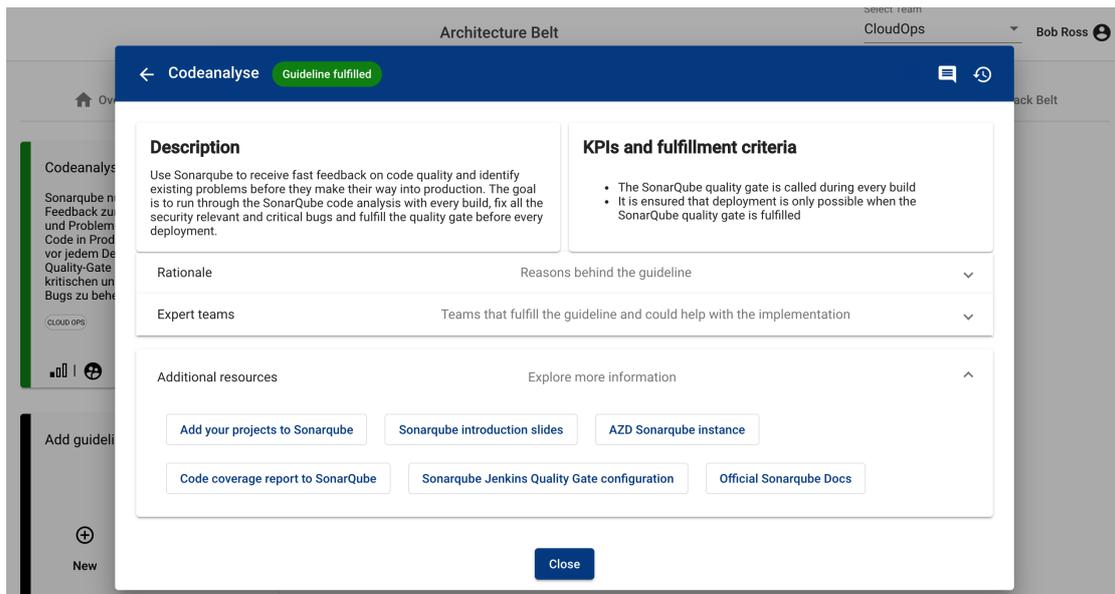


Figure 6.4.: Guideline detail view dialog of a specific guideline

The goal of the guideline detail screen dialog is to present more information about a specific guideline, additionally to the information presented to the user in the team-specific guideline overview screen. For clarity and navigational purposes, the title of the guideline is displayed again. Additionally, the full description of the guideline is shown as well as the solution criteria for a guideline, aiming to give more insight into how to fulfill the guideline. More information can be accessed by expandable components which include the following contents:

- A **rationale**, detailing the reasons behind a guideline and giving additional context, aiming to build a better understanding of the underlying motives and therefore increasing acceptance.
- **Additional resources**, which are links that can redirect to more information, e. g. to Wiki-pages, example code, reference implementations etc. These resources

consist of a description which is shown to the user and a corresponding link that is opened in a new tab when a user clicks the respective additional resource.

- **Expert teams**, which are teams that fulfill the guideline and could help other teams with the implementation. In contrast to the rationale and the additional resources, these teams are not specified by a user creating or changing the guideline, but they are automatically calculated and displayed. In the current state of the prototype, teams that fulfill a guideline automatically become expert teams and are displayed with the date on which they achieved the fulfillment of this particular guideline. Due to the fact that this could lead to a large list of expert teams quickly, only three expert teams are shown at once. In the future, a more sophisticated way of identifying expert teams could be useful, as explained in section 6.7.

The top bar of the dialog includes the current guideline status next to the guideline title and also two additional buttons on the top right side. The button with the clock history icon leads to the **guideline history**, which shows logs on when guidelines have been set to fulfilled or not fulfilled by which user. This feature gains in importance once automated tests for guidelines get integrated because it could then be used to show the times and results of automated tests that were running to determine the fulfillment status of a guideline. At the moment, it displays a history of status changes of the guideline, that means if the guideline was set to fulfilled or not fulfilled, with each entry including a time stamp, the type of status change as well as the responsible user for the change. With certain guidelines, especially the ones based on legal requirements, this can be important for auditing as well. The comment button gives the teams the **possibility to enter a team specific solution and rationale for their guideline**. This can be valuable when the team has reasons not to fulfill the guideline as intended by the guideline specification. Documenting these deviations from the standard can help to collect valuable feedback on refining guidelines later on. If there is an existing team-specific solution or rationale for the guideline, it is then shown in the guideline detail screen so that the team is reminded of how they are fulfilling the guideline.

6.3.3. Team dashboard

The team-specific dashboard is the home screen that is shown to a user. It prominently displays the current belt of the selected team and the percentage of the progress to the next belt that can be achieved. The activity feed on the right side of the dashboard displays cross-team information. At the moment it displays the achievements of a new belt by a team with the associated date of that event. This could be extended in the future by also presenting other collaborative information, such as welcoming new

6. Implementation

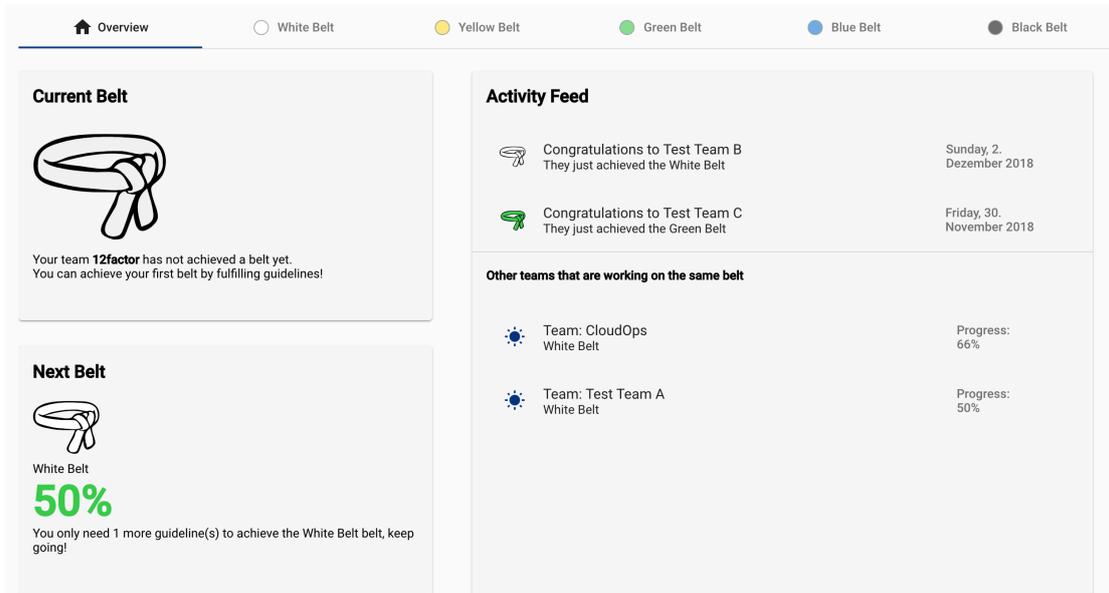


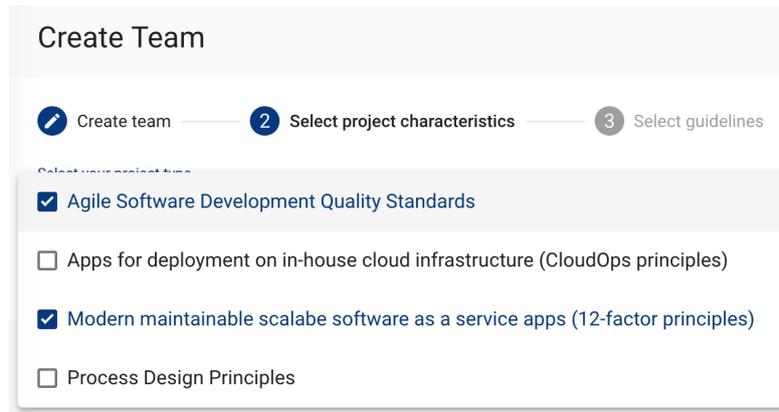
Figure 6.5.: Team-specific dashboard displaying the current belt, activity feed and the progress of other teams

team members or displaying team-specific information, like the new fulfillment of a guideline. The area below the activity feed shows other teams working on the same belt as the current team of the user and their progress in the belt, aiming to further motivate teams, but also enabling them to identify teams who are facing similar challenges at the moment or who are in a similar state of project maturity. This could spark higher collaboration and exchange between those teams.

6.3.4. Team creation and guideline mapping

The team creation feature allows users to easily add new teams and add existing or new users of the application as members to that team. Furthermore, during the creation process, it is possible to select from predefined project or team characteristics as illustrated in Figure 6.6. These project characteristics are linked to certain guideline sets, which are based on the inclusion and exclusion of all guidelines of a single tag or the combination of tags as well as the inclusion or exclusion of specific guidelines.

Depending on the selection of the user creating the team, the application pre-selects the relevant guidelines in the guideline selection step that can be seen in Figure 6.7. Nevertheless, the user can still adjust the selection based on the project or team requirements. As described earlier, new guidelines can also be added later to a team.



The screenshot shows a 'Create Team' interface with a progress bar at the top indicating three steps: 1. Create team, 2. Select project characteristics (current step), and 3. Select guidelines. Below the progress bar, there is a section titled 'Select your project type' with four checkboxes:

- Agile Software Development Quality Standards
- Apps for deployment on in-house cloud infrastructure (CloudOps principles)
- Modern maintainable scalable software as a service apps (12-factor principles)
- Process Design Principles

Figure 6.6.: Selection of project and team characteristics when adding a new team

Guidelines then get pre-selected accordingly to the chosen characteristics.

6.3.5. Team and guideline statistics

The fourth core feature in the current prototype of the application is a team and guideline overview, which allows the user to get an overview of his teams and the fulfillment status of the selected guidelines. This allows for easy comparison and identification of existing problems.

6.4. Features and possible extensions based on social design principles

Since the value of the presented tool is dependent on the active usage and contribution of its users, we incorporate insights from social science and gamification into the design of our features. Even if it is important to integrate the tool-support into a broader scope such as a community and collaborative process, as presented in our approach, it is possible to use certain design principles of online communities to encourage desirable behavior [96]. In the following, we list selected design claims based on the research by Resnick and Kraut [96]. The goal of this section is to describe how some of the selected, relevant design claims have already been taken into account with already implemented features, and to suggest ideas how others could be used in possible future features. The design claims presented could not only improve the level of participation and contribution, but also lead to better compliance with guidelines. All presented design claims are drawn from the chapter "Encouraging Contribution to

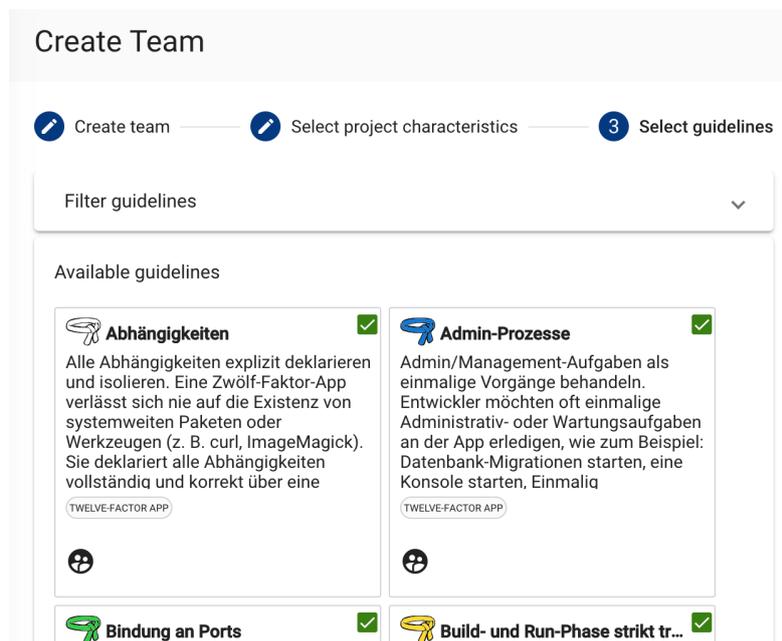


Figure 6.7.: Pre-selection of guidelines based on the chosen team or project characteristics

6. Implementation

Select Teams ▼

User Specific Overview ^

... for Teams you're team owner ... for Teams you're member in

Team	Achieved Belt	White Belt		Yellow Belt				Green Belt	Blue Belt		Black Belt		
		Abhängigkeiten	Codebase im Veralt.	Build- und Run-Pha.	Dev-Prod-Vergleich.	Konfiguration	Prozesse	Unterstützende Dia.	Bindung an Ports	Admin-Prozesse	Einweggebrauch	Logs	Nebenläufigkeit
12factor Newbies		+	+	-	-	-	-	-	-	-	-	-	-
12factor Ninjas		+	+	+	+	+	+	+	+	+	+	-	-
Partly 12factor		○	+	+	○	-	-	+	-	-	-	○	○

Legend:

-  fulfilled
-  not fulfilled
-  not associated with team

Figure 6.8.: Overview of teams and guidelines and their respective fulfillment status

Online Communities" [96] and resolve around encouraging contribution in an online community, because this has the highest priority in our case. These design principles are not only relevant for an online community, but many of them originate from social psychology lab experiments and general findings from social science, which makes them also valuable in general for our community approach. In the future, other categories of design principles could be taken in account as well, e. g. encouraging commitment or dealing with newcomers. The numbering of the design claims represents the original numbering by Resnick and Kraut for quicker identification within their book chapter.

"Making the list of needed contributions easily visible increases the likelihood that the community will provide them." (Design claim #1 in [96])

The guideline overview for a team is designed to quickly and easily show the members of a team which guidelines have already been met and where action is needed within the team. The design principle could also be kept in mind at a later stage when extending the application, e. g. by clearly listing the guidelines that need further specification or improvement by the community and how and where exactly a member of the community can contribute. This could also fit to prompting the users of the application to contribute to the community by rating existing or proposed guidelines as well as giving qualitative feedback to improve those guidelines.

This also fits well with the second design claim, which states that

"Providing easy-to-use tools for finding and tracking work that needs to be done increases the amount that gets done." (Design claim #2 in [96])

Therefore, the application could provide a small task list that summarizes the need for certain guideline improvements that are required, e. g. extending the guideline description, providing a more comprehensive rationale or specifying KPIs or solution criteria for the guideline.

"Compared to asking people at random, asking people to perform tasks that interest them and that they are able to perform increases contributions." (Design claim #3 in [96])

This design principle could be implemented by tailoring the call for contributions stated in the previous two design claims to specific teams that are most suitable for the tasks. This could be done by asking only those teams to vote on guidelines that are categorized with tags that also fit the project characteristics of the respective team. Furthermore, call to action for refinement or validation of guidelines could be displayed mainly to expert teams of those guidelines.

6. Implementation

"People will be more likely to comply with requests if they come from others who are familiar to them, similar to them, are attractive, are of high status or have other noticeable socially desirable characteristics." (Design claim #11 in [96])

The familiarity between different teams based on similar project characteristics and selected guideline tags could be used to recommend guidelines to each other. Also, it might be considered to introduce a form of a reputation or endorsement system where teams and/or individual users can endorse each other for contributions to the community and thereby achieve a certain rank or status, which could then, in turn, lead to a higher level of compliance to the requests from the community.

"People are more likely to comply with a request when they see that other people have also complied." (Design claim #12 in [96])

This design claim fits well with the news feed and belt-specific progress leaderboard displayed on the team dashboard page, displaying teams achieving new belts and how far they are progressing in the belt that is relevant for the current team. Furthermore, the expert team feature shows teams who have been complying with a guideline for a certain time period.

The news feed could be extended to also include information on how teams or individual users are contributing to the community, e. g. by adding new guidelines or refining existing ones. Also, aggregated statistics on specific guidelines might be a good way to further benefit from this design claim, such as revealing an information like "87% of teams are fulfilling this guideline" on a certain guideline, which could further encourage non-compliant teams to comply as well.

"Providing members with specific and highly challenging goals, whether self-set or system-suggested, increases contribution." (Design claim #13 in [96])

The presented tool-support challenges teams to reach a higher level until they reach the black belt. This could be further enhanced by providing other forms of goals such as badges or achievements. A team can also self-set goals by adding their own, team-specific guidelines.

"Coupling goals with specific deadlines leads to increases in contribution as the deadline approaches." (Design claim #14 in [96])

This design principle might become relevant in the future within additional features. One potential aspect is to limit the call for participating in voting or providing feedback to specific time periods. This would be similar to the process that was observed in

6. Implementation

the case study during the "request for comments" phases, in which participants of the community have a limited time period to give their feedback on a specific guideline within a community board, before the process continues.

"Goals have greater effects when people receive frequent feedback about their performance with respect to the goals." (Design claim #15 in [96])

For this purpose, the team dashboard always indicates the current progress in the current belt on the way to the belt that can be achieved next. In the future, more detailed feedback could be given to the user, e. g. regarding their progress on certain badges or other achievements. But not only feedback on the progress could be given to the user, but on actual performance, as stated in the design claim, e. g. by using KPIs for guidelines - where possible - and displaying the achieved value for that KPI.

"Combining contribution with social contact with other contributors causes members to contribute more." (Design claim #16 in [96])

This design claim highlights the necessity to integrate the tool into a broader context, as stated earlier. By including the tool in existing processes and collaboration, especially the ones focused on personal, face to face interaction, the awareness for the tool and actual usage could increase sharply. The tool could be included in existing communities of practice, sprint retrospectives or during the assessment of minimal viable product iterations (e. g. to demonstrate a high fit with the organizational strategy and cross-team guidelines to the management).

"Performance feedback - especially positive feedback - can enhance motivation to perform tasks." (Design claim #18 in [96])

Similar to design claim #15, we rudimentarily implement this design claim by showing the teams their progress in a current belt and encouraging them to achieve compliance with further guidelines. In the future, the design claim could be used to a greater extent, e. g. by alert or notification messages with congratulations after reaching a new belt, badge or other achievements. Also, the newsfeed within the dashboard could incorporate more positive feedback.

"Site designs that encourage systematic, quantitative feedback generate more verbal feedback as well." (Design claim #19 in [96])

By promoting to give feedback on a Likert scale on how valuable the guideline is for a specific team, as illustrated in the screenshot in Figure 6.11, the goal is to increase the amount of more detailed feedback and reasoning behind the rating as well.

"Comparative performance feedback can enhance motivation, as long as high performance is viewed as desirable and potentially obtainable." (Design claim #21 in [96])

"Performance feedback, especially comparative performance feedback, can create a game-like atmosphere that may have undesirable consequences in some communities." (Design claim #22 in [96])

The progress indicator on the dashboard presenting other teams working on the same belt already aims to provide comparative performance feedback as described in design claim #21, and can be seen as a sort of leaderboard. The application could also be extended by a more comprehensive leaderboard, listing teams from all belts and other statistics. Additionally, once KPIs for guidelines are implemented, the comparative performance feedback could be done in even more detail. Nevertheless, as design claim #22 warns, such measures could also have undesirable effects. This is especially the case in supportive and learning environments [96], in which community members might prefer an environment that builds on positive, supportive aspects instead of competition, which could also lead to more people "gaming the system" to achieve progress and outperform the competition.

"Rewards, whether in the form of status, privileges or material benefits - motivate contributions." (Design claim #23 in [96])

Achieving a new, higher belt can already be seen as a sort of status, but this could be further extended with other kinds of achievements, e. g. badges. Furthermore, depending on how the tool gets embedded in a broader context, it would also be possible to think about including privileges or material benefits.

6.5. System architecture

The following section provides a brief insight into the technical architecture of the application.

The system architecture is kept very simple. It consists of the Angular single-page-application serving as the frontend for the user, which is communicating with the Spring Boot backend and its exposed REST-API via HTTP calls. The Spring Boot backend then accesses a MySQL database with the use of JPA⁹ and Hibernate ORM¹⁰. Figure 6.9 illustrates a high-level overview of the architecture.

⁹ <http://spring.io/projects/spring-data-jpa>

¹⁰ <http://hibernate.org/orm/>

6. Implementation

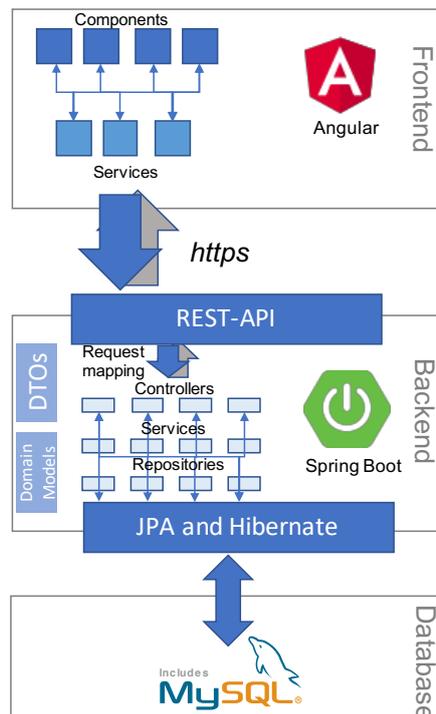


Figure 6.9.: High level overview of the system architecture

The frontend architecture is closely aligned to the architecture¹¹ intended by the Angular framework. Two of the basic building blocks in the Angular framework are components and services. Both are simply classes, with decorators that indicate their type and provide the necessary meta-data that tells the Angular compiler how to handle them. Components control the different parts of the application. A component provides a reusable view, consisting of an HTML template, a stylesheet as well as the necessary logic. Services, in general, provide functionality which is not directly related to views. In our case, this is mainly the communication with our REST-API via the Angular HTTP client, which is based on the XMLHttpRequest interface exposed by browsers¹², as well as the communication between various components in cases where input and output component bindings are not feasible, because the data would have to be passed through multiple components. Another basic building block of an Angular application are modules. Modules can include a set of components and services and help to organize code into distinct functional units which can be very helpful in developing complex applications. So far, the web application is kept in one root module due to its rather small size, but with further extensions, it could quickly become advantageous to split the application into multiple modules.

6.6. Class diagram

The following class diagram presented in Figure 6.10, based on the Unified Modeling Language (UML) notation specification [86], presents the main elements of the application. For a better overview and readability reasons, attribute types are omitted, as well as reoccurring attributes that are the same for every class, namely "ID", "last updated" and "created at".

The guideline definition represents the guidelines that exist in the overall guideline pool of the application. It consists of information for the respective guideline that is the same for each team, e. g. the title, description, rationale, etc. Furthermore, a guideline definition belongs to a certain belt and multiple tags can be assigned to a guideline. The actual guideline class represents an instance of a guideline that belongs to a certain team. A team can have multiple guidelines and a guideline definition can be used by multiple teams. For each guideline definition a team is using, a guideline instance gets created to store the current fulfillment status as well as other additional information, e. g. a team specific rationale and solution that the team can document in the case that they would like to fulfill or use the guideline in a different way than the guideline definition originally specified in the guideline definition. For a guideline, we do not use

¹¹ <https://angular.io/guide/architecture>

¹² <https://angular.io/guide/http>

6. Implementation

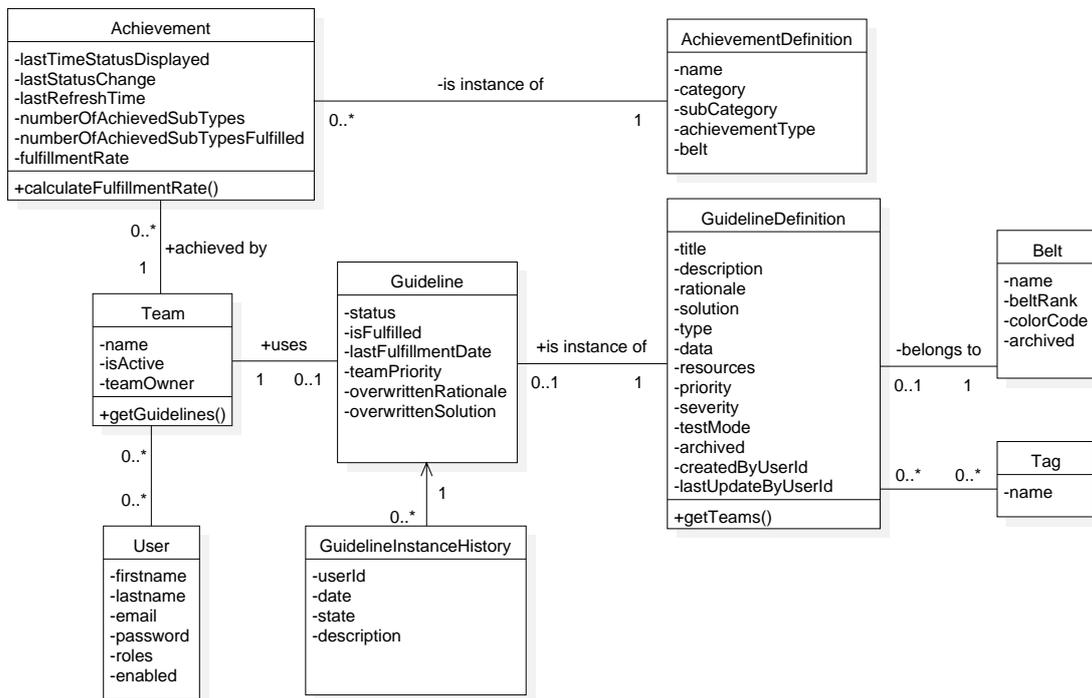


Figure 6.10.: Underlying class diagram of the application

a regular auto-generated ID, but a composite ID as a primary key which is composed of the team ID and the guideline definition ID, because every guideline instance can be uniquely identified by its associated team and original guideline definition. The guideline of a team can also have multiple history entries which store information when and by whom a certain fulfillment status change has been conducted. Lastly, multiple users can be assigned to a team and a team can accomplish multiple achievements. At the moment, the only achievement type is the belt achievement, which describes that the team progressed into a higher belt. Nevertheless, for further extensions, the existing model can be used to introduce additional achievement types, e. g. badges.

6.7. Possible extensions and next steps

Even if the current state of the application is already sufficient for usage in the case study company, there are a large number of possible further extensions of the application. Some of the possible next steps are described in the following.

Offer a rest endpoint for quality gate checks from build and deployment pipelines

The web application could be extended to expose an endpoint that can be called within a build script by a build tool (e. g. Jenkins) as a quality gate, next to the other forms of testing that are usually performed within a CI/CD pipeline (e. g. end-to-end tests). The response of the web application indicates whether all necessary mandatory guidelines for a deployment are fulfilled. The build and deployment would continue if the response is positive, in case of a negative response the deployment would fail. This is similar to how the integration of other tools works, e. g. how earlier versions of the static code analysis tool SonarQube could be used to break the build¹³.

For the request, a standard GET request could be used, with an URL parameter identifying the team (e. g. a unique team name or ID). With this information, the web application can check whether all necessary guidelines are fulfilled or a sufficient belt is reached. For usability purposes, the frontend could offer a code snippet for teams that already includes their identifier and can easily be copied into the build script. A more sophisticated future version could offer a dedicated plugin for common build tools.

Automated testing of guideline fulfillment

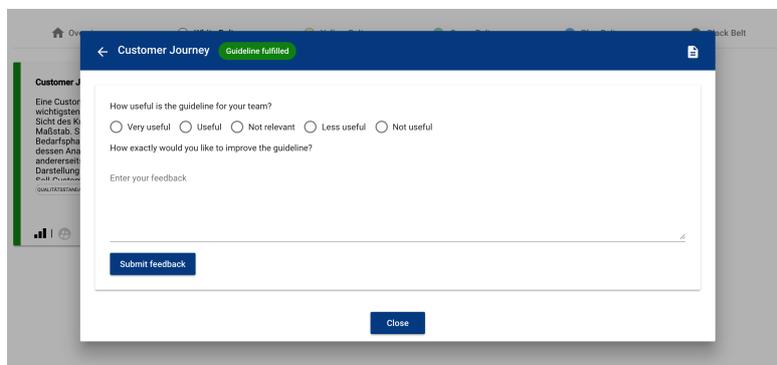
Instead of relying solely on the manual effort of agile team members or architects to maintain the current fulfillment status for the different guidelines, the goal is to

¹³ <https://docs.sonarqube.org/display/SONARQUBE53/Breaking+the+CI+Build>

automatically check the compliance with guidelines, where possible. In the UI, a small lock or a similar icon on a guideline could indicate that the guideline fulfillment status cannot be changed automatically, but is determined by an automated test running in the background. For the purpose of testing, the application could be extended by an integration controller and service in the backend that collects data from various sources that are relevant to the guideline fulfillment (e. g. from static code analysis tools, configuration management databases, performance metrics from the cloud infrastructure etc.). A testing service would then run the actual tests based on the acquired data. To identify what needs to be tested to determine the fulfillment status of the guideline, it would be helpful to allow the definition and management of KPIs for guidelines first, which then provide a measurable value that should be automatically testable in some cases.

Feedback for specific guidelines

To support the handling changes phase of the approach, it is important to collect relevant feedback from all involved stakeholders. The possibility to quickly give feedback for a specific guideline could facilitate a higher amount of feedback and more target-directed feedback. By combining measurable feedback on a Likert scale with qualitative feedback, the feedback can both be aggregated for statistical insights, e. g. how well a guideline is perceived by the target stakeholders, but can also provide qualitative insight into how a guideline could be improved. The web application could



The screenshot shows a feedback form within a web application. The form is titled "Customer Journey" and "Guideline fulfilled". It contains two questions: "How useful is the guideline for your team?" and "How exactly would you like to improve the guideline?". The first question has five radio button options: "Very useful", "Useful", "Not relevant", "Less useful", and "Not useful". The second question has a text input field labeled "Enter your feedback". There are "Submit feedback" and "Close" buttons at the bottom of the form.

Figure 6.11.: Possibility to provide feedback for specific guidelines

be extended to store a guideline maintainer for each guideline and the feedback could then be send to that guideline maintainer. It could also be displayed as comments on the guideline if transparent feedback is preferred. Furthermore, a new statistics page could be introduced that aggregates and visualizes the feedback on guidelines. This

should provide a good basis for the community to discuss, review and adjust existing guidelines or for guideline maintainers to make minor tweaks to the guideline.

Export feature for audit-compliant documentation

For audit requirements, it might be necessary to be able to export some documentation on the guideline compliance that, amongst other information, includes the links to the artifacts that are necessary for the guideline fulfillment. This documentation then has to be stored safely for a pre-determined time frame, often over multiple years. For agile teams and other stakeholders, it could be very valuable and time-saving when they do not have to create those documents for auditing reasons manually anymore, but if they could instead rely on the application to automatically export the necessary information.

7. Evaluation

In this chapter, we describe the evaluation of the resulting artifacts of this thesis. In the first section, we describe the goal of the evaluation and how we conduct the evaluation. In the sections thereafter, we present the actual results of the evaluation. Section 7.2 presents the evaluation of the collaborative approach, Section 7.3 demonstrates the results of the evaluation of the tool support.

For the evaluation, semi-structured, qualitative, semi-structured expert interviews with fifteen participants have been carried out. The participants all belong to the case study organization and are active in roles that are relevant for the scope of the developed artifacts, mainly enterprise architects and agile developers.

7.1. Goal and methodology

In this section, we describe the approach of the conducted evaluation.

For the evaluation of our two artifacts, we use the means of our case study. Within a case study, interviews can be a suitable source for information gathering for the evaluation of information systems artifacts [26]. Based on the evaluation method types in design science research presented by Peffers et al. [88], our evaluation methodology could be categorized as a combination of "Prototype", which is described by Peffers et al. [88] as "Implementation of an artifact aimed at demonstrating the utility or suitability of the artifact", "Expert Evaluation", which is summarized as an "Assessment of an artifact by one or more experts" [88] and "Case Study", which is the "Application of an artifact to a real-world situation, evaluating its effect on the real-world situation." [88].

The goal of the evaluation is, first of all, to assess whether the artifacts are perceived as valuable by key stakeholders and potential users, and second, to identify and provide insights for further refinements of the developed solution artifacts.

To achieve this, we conducted semi-structured interviews with experts within the organization based on an evaluation questionnaire we developed in multiple iterative steps. The questions of those semi-structured expert interviews are listed in Appendix A.1.

For the purpose of reaching the goal of the evaluation, we demonstrated the key results of our research and presented the two resulting solution artifacts during the interviews, followed by the interviewees answering the questionnaire.

7. Evaluation

Table 7.1 lists all interviewed experts, their main role and their professional experience in years in enterprise architecture and agile development. As presented in the table, on average, the interviewed enterprise architects have a wealth of experience in agile development, which makes them especially suited for the purpose of the evaluation. Most of the enterprise architects and some of the agile developers would have even more years of experience in software development if traditional development methods are taken into account. But due to the nature of this thesis, with the main focus being on agile and lean environments, we list only their agile development experience. Furthermore, for comparison and simplicity reasons, we categorize a participant with the main role of being a solution architect as an enterprise architect due to the closely related tasks and activities carried out in comparison to the other interviewed enterprise architects. The role of the enterprise architects also varies in detail, with some enterprise architects focusing more on development tasks and being closer to agile teams as others. For the purpose of our evaluation, it was important to get a diverse impression and opinions from different perspectives, hence the balanced distribution of interviewed roles.

No	Alias	Main role	Professional experience in years	
			Enterprise Architecture	Agile Development
1	EA1	Enterprise Architect	2	5
2	EA2	Enterprise Architect	1	10
3	EA3	Enterprise Architect	1.5	6
4	EA4	Enterprise Architect	3	6
5	EA5	Enterprise Architect	5	15
6	EA6	Enterprise Architect	2	-
7	EA7	Solution Architect	6	10
8	EA8	Enterprise Architect	2	5
9	AD1	Agile Developer	-	7
10	AD2	Agile Developer	-	1
11	AD3	Agile Developer	-	3
12	AD4	Agile Developer	-	1
13	AD5	Agile Developer	1	3
14	AD6	Agile Developer	-	7
15	AD7	Agile Developer	1	3

Table 7.1.: Evaluation interview partners

7.2. Evaluation of the collaborative approach

This section presents the evaluation results of the collaborative approach, introduced in Chapter 5, in detail. For this purpose, we present the respective question of the evaluation questionnaire, the quantitative results for each question as well as a summary of the qualitative answers that were given by the participants. In the last part of this section, we summarize and provide an overview of the results of all questions evaluating the collaborative approach.

7.2.1. Value for agile teams and enterprise architects

Introduction in the questionnaire: "The approach focuses on creating a community of developers and solution architects from different agile teams as well as enterprise architects, meeting regularly, discussing and deciding on architecture principles and guidelines together."

Question 1: In your opinion, how valuable is such a collaboration for agile teams?

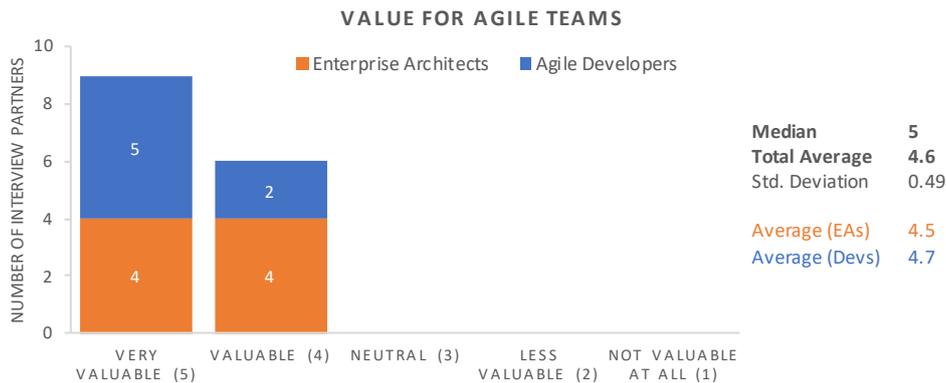


Figure 7.1.: Evaluation results for question 1

The close collaboration between agile teams and enterprise architects, as suggested in the collaborative approach, is rated as very valuable by the interviewees. The agile developers rate the value for themselves slightly higher as enterprise architects consider the value for agile teams.

One of the main advantages of the approach which is identified and mentioned by multiple participants is that the approach facilitates a better understanding (EA1, EA2, AD3, AD4, AD5). The interviewees specifically refer to a better common understanding (AD5), a more conclusive understanding of the global idea of architecture in an

organization (AD4), a better understanding of guidelines (AD3) and better access to information and guidance (EA1). The approach also provides a better possibility to get insights into topics developers usually do not focus on (AD5). Teams mostly concentrate on reaching a local optimum, but do not necessarily know the overall context (EA7). Other commonly stated themes are the better, shorter feedback cycles and the regular exchange, which can lead to continuous improvements of existing guidelines (AD5, AD3, EA3) and gives a better impression "what works and what doesn't" (EA3). Teams can then save time through good guidelines and can focus more on their development activities (EA6). EA5 stresses that "architecture won't work without input from those which have to implement it at the end". Such a close collaboration as suggested in the approach could significantly increase acceptance within the teams (AD6, EA8). EA4 sees a better awareness of principles and guidelines, ensuring that teams do know about them and do not ignore guidelines during their work, which otherwise can cause problems like delays and higher costs, e. g. by not being allowed to deploy to a productive environment. The closer collaboration not only between the enterprise architects and agile teams, but also between the agile teams themselves, can contribute to the development of products that are more homogeneous and fit together more seamlessly (EA7). Furthermore, teams can profit from the experience of others and not make the same mistakes (EA6). There is also a high value because of the better, more joint cross-team direction (EA4) and increased speed through direct communication and feedback (EA2).

AD1 stresses that representatives from other relevant involved stakeholders should also be included (e. g. operations, IT-security, data protection). AD2 states that even if both sides can profit from the collaboration, it is also associated with more effort. Teams could also be worried that they lose some freedom because of guidelines, but on the other side, they get a say in the architectural topics (EA6).

Question 2: In your opinion, how valuable is such a collaboration for enterprise architects?

Similar to the first question, the value for enterprise architects is also considered very high. Agile developers estimate the value for enterprise architects slightly lower than enterprise architects rate the value of the collaborative approach for themselves.

The value for the enterprise architects is seen by EA4 in "dissolving the ivory tower, more direct contact with the teams, being closer to the results and how principles and guidelines affect the results". This statement fits well to some of the points that are also listed by multiple other interviewees: The approach could help architects to learn that top-down processes not always work (EA6) and that "one architect can't know everything, people who implement it might come up with better or more pragmatic

7. Evaluation

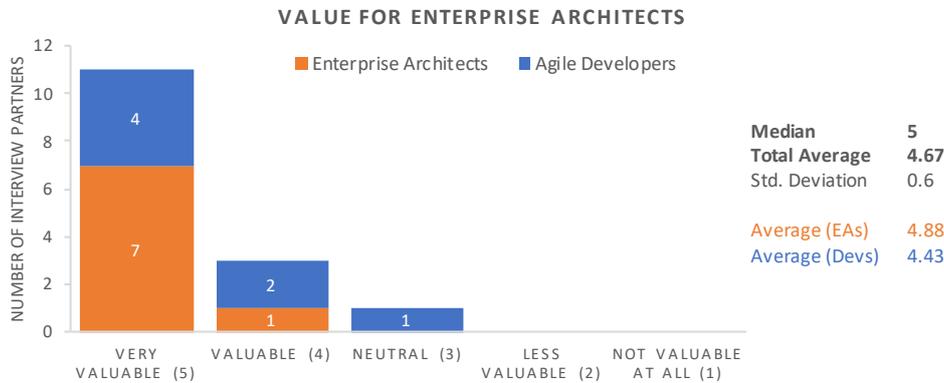


Figure 7.2.: Evaluation results for question 2

solutions" (EA5) and prevent "architecture in the ivory tower" (AD7). They can establish close contact with the teams, which is often not the case, and they are closer to the actual issues (EA6, EA7, EA8). It can also help architects to better understand the worries and challenges of developers (AD5). For architects, it is important to know the problems of "the base" (AD4).

More traditional groups (e. g. the committees) often lack the feedback what impact the decision actually had after they were made and applied during implementation (EA6, EA8). The value in the approach, on the other hand, is seen in the ability to receive faster and better feedback of agile teams (EA2), as well as direct feedback to guideline proposals, so that adjustments can be discussed together immediately (AD6). These better feedback loops and the closer collaboration could lead to a better applicability of guidelines (EA3), because the input of developers shines light on the aspect whether certain guidelines even can be implemented at all (AD3) and whether they actually work in practice (AD7). It also allows architects to adopt guidelines from teams that might already use their own team-specific guidelines (AD5). As a result of the collaboration and letting teams join in on the process, they can spark more motivation in teams (EA1), increase awareness and therefore compliance because "rules must be known in order to be followed" (EA4). Finally, the approach would also facilitate better transparency as it structures and centralizes collaboration (EA1) and because of the feedback cycles and KPI measurements (EA1, EA2).

7.2.2. Challenges

In contrast to the previous questions, the following question is of a purely qualitative nature. Therefore, there are no quantitative results.

Question 3: What challenges do you see in the actual implementation of such a community? In your opinion, which problems need to be considered and solved in order to make the collaborative approach successful?

The first of the identified challenges by the expert interviewees is the necessary organizational shift and cultural fit. EA4 states that the presented approach could be especially challenging for organizations with a structure close to traditional line organizations (very hierarchical, with the line of command being carried out from top to bottom). An approach like the presented one requires that the top and middle management have to be willing to give up power and fully commit to the new structure (EA4). A similar point is made by AD2, who explains that architects have to be open to listen to the developers and let them influence their decision making (AD2), which might result in less power and authority of the architects. Similar to this opinion, EA8 notes that both agile teams and enterprise architects have to be open for approaching each other and there must be a common willingness for change. EA4 acknowledges that this requires a change process and that it may take some time. EA6 also assesses that the approach, in general, would need time to be able to provide value. Regarding the organizational and cultural aspect, EA3 puts emphasis on the "people" aspect and sees the challenge on how to build the community in a way that the participants get to know each other as people and not by their roles (which may lead to the tension of "police" vs. developers), so that collaboration on a constructive and open level can take place. Since communities build on the passion and contribution of its members, the organizational culture has to encourage such a community so that enough participants are available (AD7). This also means that members of agile teams is given the time by the management, product owners, etc., to participate in such a community (EA7). A common challenge in these communities is that potential participants prioritize the meetings that are mandatory for them (e. g. planning meetings, retrospectives, grooming etc.). That is why it might be challenging if there is no mandatory aspect to the community (EA7).

The second challenge is the potential conflict of interest. For example, acceptance at the beginning of the community establishment could be a problem. Agile teams would have to invest a part of their time, but it does not immediately help them to reach their project or product goal, which has priority for them (EA6). Furthermore, developers usually tend to prefer fewer guidelines because the resulting restrictions and additional effort might "annoy" them (AD3). This could lead to developers blocking new guidelines or removing existing ones, as well as to developers and architects not finding common ground (AD3). In general, architects and agile teams might have conflicting goals, so there has to be a clear way and rules how decision making can be achieved anyways (AD1). Therefore, it has to be clear for both sides that the

common goal is to find pragmatic solutions that are acceptable for both sides (AD6). Furthermore, a common definition and consistent use of important recurring terms to facilitate a better understanding between agile teams and enterprise architects can be helpful (EA8).

The third major challenge is scalability. The question regarding scalability is how the collaborative approach could be implemented with a high number of teams (EA6). AD5 notes that in the case of a high number of teams, it is challenging to introduce overarching, uniform guidelines. The size of the community could become a problem, because "if you send one representative for each team to join the community, the group will become very large and possibly inefficient" (EA5). This requires that all decisions have to be made transparent and well documented for everyone who is not joining the community in person (EA6). A challenge that goes hand in hand is how to publish that information in a way that the teams understand it easily and fast. A related challenge is how to integrate new teams in a way that they don't get left behind by more experienced teams (EA6). Also, there is the need for someone to take care of creating meeting agendas and announcements to the community, because otherwise, the motivation to participate decreases (EA7).

The fourth challenge is setting the responsibilities and scope of the community. The scope has to be very clear, that means how far does the freedom and flexibility of the community go and what is "non-negotiable" (AD6). There are regulatory requirements which need to be fulfilled no matter how the community votes. The community could decide how to implement those, but the final acceptance still depends on regulators (EA5).

7.2.3. Community activities and process steps

The next questions evaluate the proposed activities of the community and the more detailed steps of the process.

Question 4: It is important that development teams do not just get architecture principles and guidelines dictated by other stakeholders, but that they are the main stakeholder themselves in managing architecture principles and guidelines and are involved in the whole approach (steps 1-7).

The main reason for agreeing with the statement above is the better acceptance and motivation of agile teams that could result from involving them more closely in the process and giving them the opportunity to have a say (AD2, EA1, EA2, EA4, EA6, EA8). EA4, for example, states that "principles and guidelines are mainly created by the enterprise architects, but the involvement of agile teams is extremely important for the acceptance and compliance by agile teams". EA8 explains that it is very important for

7. Evaluation

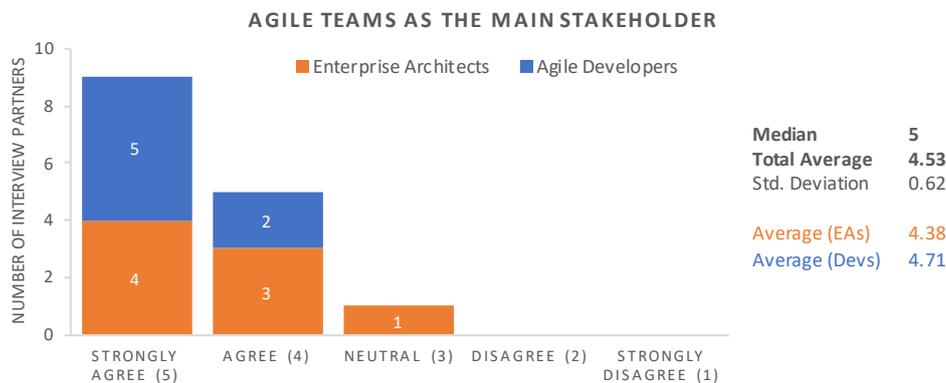


Figure 7.3.: Evaluation results for question 4

the acceptance to "turn those affected into actors". The involvement of teams also helps validate the effect of governance efforts during actual implementation (EA7, AD7) and can prevent too much time being spent on theoretical concepts (EA7). Other reasons for agreeing with the statement include the reasons already described in the previous section regarding the value of the collaborative approach for agile teams and enterprise architects.

Some interviewees partially agree due to the following reasons. Some things have to be regulated and enforced because developers do not like to do them because they mean additional effort, but they might still be necessary (AD3). Furthermore, EA5 states that "they are 'a' main stakeholder, but not 'the' main stakeholder. Finance, company strategy etc. also significantly influence architecture (EA5). Similarly, EA3 feels that the wording of the statement is too opinionated, because the main focus should not be on developers or any other role, but on the business value (EA3). This fits well with the view that agile teams should be closely involved, but clear guidance from an overall perspective is still needed (EA1).

Question 5: Agile teams can provide valuable input for creating and refining common architecture principles and guidelines drivers (especially based on their implementation experience, the issues they face on a daily basis, team-specific agreements, and their technical knowledge and insights in innovative technologies) (step 1).

There is a high agreement of the participants with the statement and the described ways on how agile teams can contribute to drivers. Issues are seen as important, because not all, but often only a little of the implementation challenges can be identified up front, even more challenging is the identification of an actual solution (EA1). Also

7. Evaluation

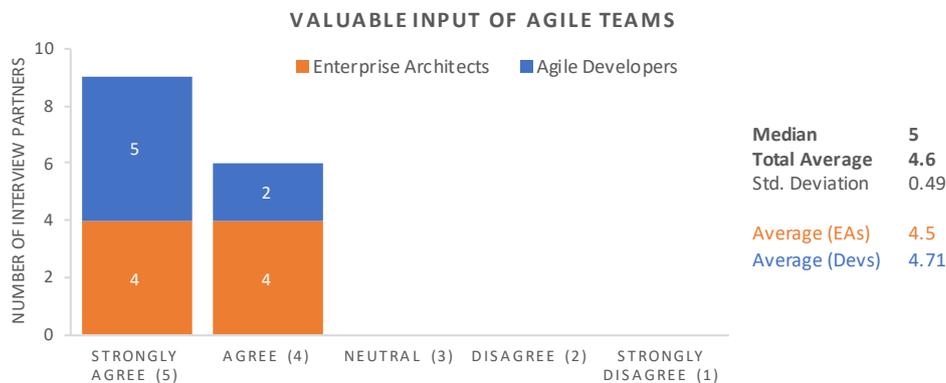


Figure 7.4.: Evaluation results for question 5

problems of the specific domain often only appear during the concrete implementation (AD6). The experience of the agile teams is seen as important as well, to achieve experience-based decision making instead of a "perfect world" theory (EA2). One developer would even like that teams should give the main input for the drivers and that enterprise architects mainly take care of the governance based on that input (AD4).

Regarding the question what other valuable input teams can provide next to the four examples listed in the above statement, a pattern could be identified regarding two additional examples of input that were mentioned more than once. Firstly, the ability of teams to provide concrete instructions, examples, code snippets or best practice implementations for others (AD3, EA5, EA6) and the possibility to form an open source community around the guidelines, thereby also taking ownership of the shared artifacts (EA5). Secondly, the ability to validate principles and guidelines and test, if they can actually be applied or if they might lead to other problems (EA4, EA6, EA7, AD7). Another point being mentioned is that teams can include feedback from end customers who use the software and might "suffer" from certain guidelines or the results and effects of certain guidelines (AD5). Furthermore, they could provide valuable experience for other teams by presenting mistakes that have occurred in a team (EA8). However, this requires an appropriate corporate culture. A similar point is raised by AD7, who stresses that in his experience, teams often face similar problems that could be solved once for all teams. Therefore the input of teams regarding these issues is import to identify and tackle those common problems (AD7).

Question 6: The input of enterprise architects to principles and guidelines is still important because of their insight into multiple teams, knowledge of business goals, strong network and their resulting ability to align multiple teams, business and IT (overall).

7. Evaluation

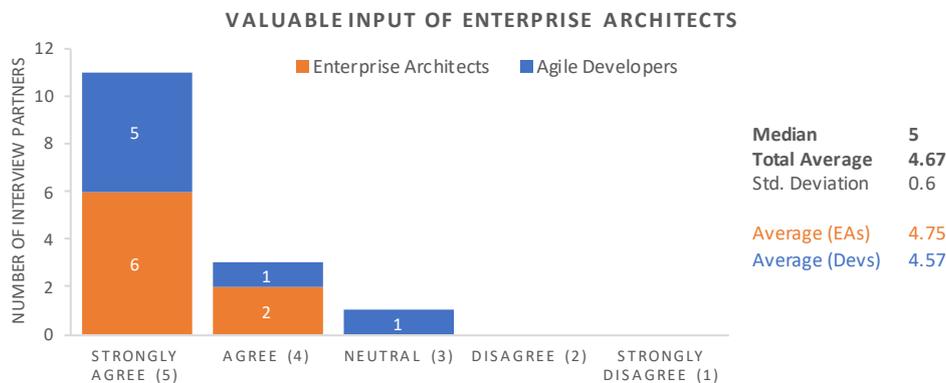


Figure 7.5.: Evaluation results for question 6

Similar to the previous question, there is again a high level of agreement from both agile developers and enterprise architects. An important advantage is the cross-sectional knowledge, which is mostly missing or not as pronounced outside of the enterprise architecture team (EA4). The network of enterprise architects can also save the teams a lot of time and effort to find suitable necessary contacts or experts within the organization (EA4). Additionally, enterprise architects are important for the communication with the management and to ensure the reporting ability, e. g. enterprise architects can select and communicate KPIs to management (EA4). Developers often do not have the business goals and business value in mind because they are too focused on technical aspects and the actual coding (AD3). This view is similar to EA6 who states that an overall perspective is important because a development team "thinks more from sprint to sprint", enterprise architects, on the other hand, can support with the long-term focus (EA6). Teams also mostly focus on solving a single problem within their scope, which might be subpar from an enterprise perspective (EA1). Another developer agrees that enterprise architects are helpful, because then teams can and should focus on their own topics, having to coordinate with other teams all the time is rather obstructive (AD6), whereas enterprise architects do not face as much operational pressure and can commit more time into certain topics (EA3). Nevertheless, there is also a view that if there is a good way for the teams to discuss and create principles, enterprise architects would be less necessary (AD4). This is opposed by the opinion that overarching topics cannot be done by the teams without losing their product focus (AD1). EA5 also agrees that enterprise architects are still needed, but stresses that they need to work in a lean and practical way and states that "Powerpoint slides which have nothing in common with reality provide no benefit and cause huge distractions" (EA5).

Question 7: It is valuable to specify generic architecture principles to one

or more specific guidelines.

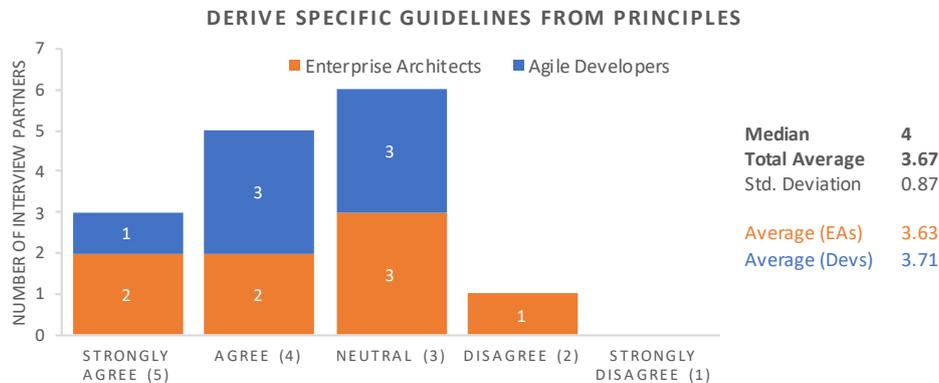


Figure 7.6.: Evaluation results for question 7

The question of whether it is valuable to transform generic architecture principles into more specific guidelines received a mixed response. On the one hand, one interviewee agrees that principles are often too generic and abstract (e. g. "reusability", "microservices", ..), and that it is very important to break principles down to something more specific and fill them with more content (EA4). Another one argues that such a specification would help to bring top-down and bottom-up approaches together (EA6). EA8 agrees with this view and stresses that specifying principles to guidelines can connect the more general, strategic top-down perspective with more concrete solutions and the bottom-up perspective. Some agile developers argue that "squishy" guidelines are useless (AD3) and that specification is important to understand what exactly the principle is about, as well as to achieve a common understanding (AD4), clarity and comprehensibility (AD2). Multiple teams might have to implement the same principle in a different way, thus guidelines can provide a better starting point for actual implementation based on the type of the problem and previous experience with the suggested implementation (EA7).

On the other hand, even if more specific guidelines give clearer guidance, they are also harder to maintain (EA1). Also, it is difficult to introduce more specific guidelines across-teams, because that specificity might not match other teams, so more general may be better in this respect (AD5). Furthermore, it depends on the way how exactly principles, guidelines and KPIs are related and organized, because the relationship is not always hierarchical and there might be many dependencies (EA2) or conflicting goals (EA6). Also, it would have to be more clearly defined what a principle is, what a guideline is and where the exact difference is (EA3, EA8). Whether something is a

principle or a guideline could also depend on the individual interpretation. EA5 argues that such a mapping can be a lot of overhead and it is hard to apply to practical work. There is the risk that people spend months to do certain mappings, which are never really used like this later on (EA5).

Question 8: It is valuable to specify one or multiple KPIs for each guideline.

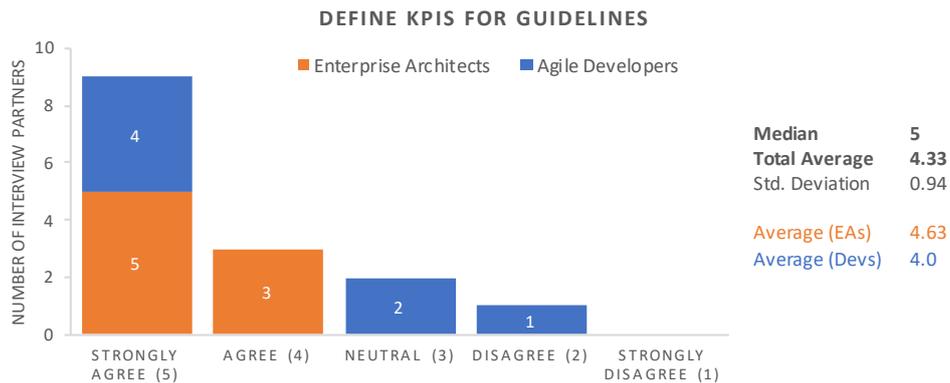


Figure 7.7.: Evaluation results for question 8

The results of this question show quite a discrepancy between the average answers of the two different roles. Nevertheless, in total, there is a large agreement regarding the questions if KPIs should be defined for guidelines. On the positive side, the interviewees value that KPIs make it measurable if a guideline is being followed or not (AD3). It also makes it possible to see and measure progress within a certain guideline, instead of only being able to tell if a guideline is fulfilled or not fulfilled (EA6). Furthermore, KPIs specify what needs to be done by the teams, while leaving them the space to find their own solution (EA1). Thereby, they can help to facilitate a better understanding of what a guideline actually means (AD7). KPIs can also provide an important data basis to get an idea of how the company is performing at the moment, but also to improve the guidelines themselves (EA4). KPIs could also be very valuable to measure the effects of guidelines themselves and if they actually have a changing impact (AD2). The ability to have measurable indicators is especially valued by the interviewees because concrete KPIs would make a more automated testing of guidelines possible (EA2, EA3, EA4, EA5, EA6, AD4, AD7). This would save time (EA6, AD4) and help to scale applying guidelines to multiple teams (EA5). Nevertheless, there are also potential risks and doubts mentioned by the participants. Defining KPIs for a guideline might not always be possible or make sense (AD5, EA5, EA8). There is also the risk with KPIs of "you get what you measure" (EA5), meaning that teams focus only on fulfilling the specific

KPIs anymore, without taking in account aspects that are important, but might be difficult to measure. Another participant is not sure how this could be realized on a large scale and questions if it makes sense from a feasibility perspective regarding the involved cost and potential value (AD6). Also, there is the concern that KPIs are used to control and restrict teams, but teams should be allowed to decide on their own to what degree the implementation of a specific guideline makes sense (AD1). There is also the challenge that KPIs should provide stable, objective measurement over time (EA2, EA4) and therefore they have to be long lasting, because if they are changed too often, they can not be used for time comparisons, e. g. over multiple years, which diminishes their value (EA4).

Question 9: It makes sense to categorize guidelines (e. g. by tags), to define target groups for guidelines and to use those tags and target guidelines later to map those resulting sets of guidelines to certain project criteria (because not all guidelines are applicable for all types of projects, but there are certain patterns of important guidelines for similar projects) (step 3).

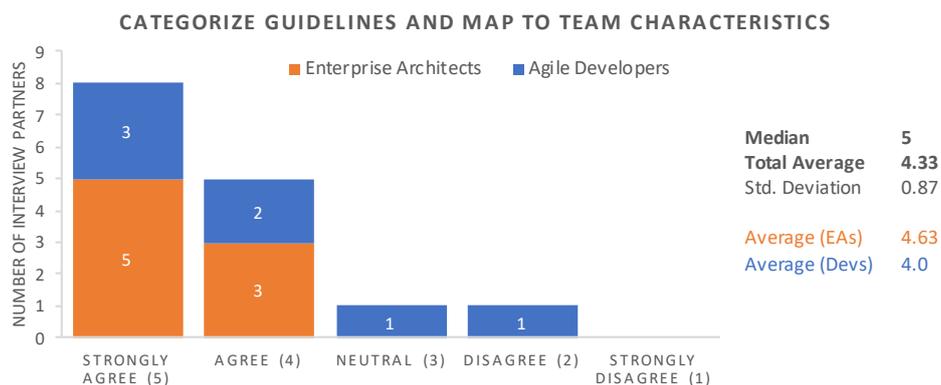


Figure 7.8.: Evaluation results for question 9

The statement is mostly supported. Positive aspects that are mentioned is that it is important to define the target group to demonstrate the relevance (EA6), enable a better, time-saving organization and navigability of guidelines (EA4, AD2, EA7) and to achieve a better fit to the respective product, project or team characteristics (EA2). Different products, projects or teams may have very different requirements and the guidelines should be adjusted accordingly (EA1, EA2, AD1, AD5) for example a system that is productive and has a lot of consumers already might have to fulfill critical guidelines more strictly than one without any productive consumers yet (AD1). Nonetheless, EA1

is not certain if the decision criteria can be simplified into a tag or target group. AD3 has a similar view and states that tagging might not be the ideal solution, but a concise statement if the guideline is important for a certain undertaking is helpful. A possible downside could be that if target groups are defined, other stakeholders will not even look through the other information, which might provide insights into cross-project or enterprise-wide information that could be valuable (AD4).

Question 10: It makes sense to let the group of developers and (enterprise) architects vote if a new guideline should be accepted into the pool or if a certain change should be made (as long as the new or changed guideline is not based on a legal requirement), instead of solely relying on a senior architect or manager to make that decision alone (step 4).

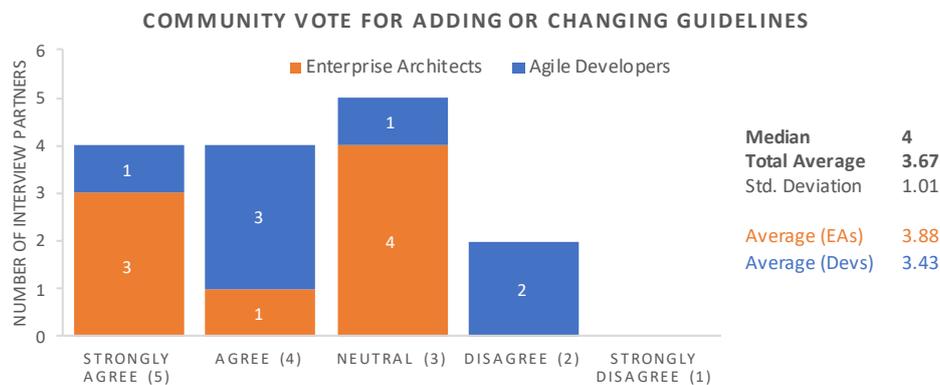


Figure 7.9.: Evaluation results for question 10

The question of whether the community should vote on certain decisions instead of having a single or a few decision makers in a lead or senior role received mixed feedback, with slightly more agreement than disagreement.

The main argument that is being brought forward on the agreement side is the higher acceptance of decisions (EA1, EA2, EA4, EA6). EA1 names the higher motivation of teams and better approval of the decisions as positive aspects. In line with the acceptance, the feeling of ownership for the guidelines could be improved as well (EA2). EA4 states that it is "not helpful if the role alone does provide a more powerful voting right", but architects would have to very clearly point out possible consequences of community decisions. AD4 is in favor of the voting because of the belief of the "wisdom of the many", leading to better decisions. EA7 argues that it is difficult to analyze all aspects of complex decisions, therefore it is very helpful to leverage the knowledge

of the community to achieve better decisions. In addition, a joint vote can prevent discussions later on in the process of applying principles and guidelines (EA7). AD7 raises the question of how good a principle or guideline can be when the majority of agile teams are against it. Nevertheless, he also sees the possibility of decisions that might receive a lot of disapproval by agile teams, although they are important for the future of the organization (AD7). AD5 likes the idea in general, but notes that such a voting approach might be difficult to implement with a large number of teams and team members. Similar to this view, EA4 raises the question if a voting mechanism is feasible in practice, because it means additional effort, which might not be possible in some organizations.

A challenge with the community voting is how mandatory guidelines should be handled and how they can be separated from other guidelines (EA2), because legal requirements and top-level management decisions have to be taken in account and usually cannot simply be overruled (EA4). There is also the question if not only legal aspects should be excluded from the vote, but also security guidelines (EA6). It also has to be ensured that useful and important guidelines are not being blocked by the participants, so the maturity of the community and participants has to be high enough (AD6, EA6). The introduction of new guidelines could also mean a lot of effort, e. g. because of necessary changes to existing architecture, which is why developers might also try to block new guidelines (AD3). Therefore, there should be more focus on an open discussion instead of an actual vote (AD3). AD2 agrees with the general idea of voting, but believes that there should be someone in the end to take over the responsibility of the decision and who can use some sort of veto power. AD7 has a similar point of view and sees the necessity of decisions driven from an organizational perspective, but also stresses that it is important to get an idea on how difficult the introduction of certain principles or guidelines would be because of the potential disapproval by agile teams. EA5 adds that an acceptance criterion for guidelines should be a practical implementation with at least two teams or technologies, wherever possible.

Question 11: It is important that enterprise architects support agile teams in applying architecture principles and guidelines (step 5).

This proposition received strong support from both groups, with developers agreeing especially strongly. The main argument for the agreement of developers is that enterprise architects should not only introduce theoretical guidelines, but make sure that teams can actually use them in practice (AD5) and work together with the agile teams to develop a specific implementation (AD3). Since some guidelines might be more complicated to implement, support can be valuable (AD6). The close collaboration is

7. Evaluation

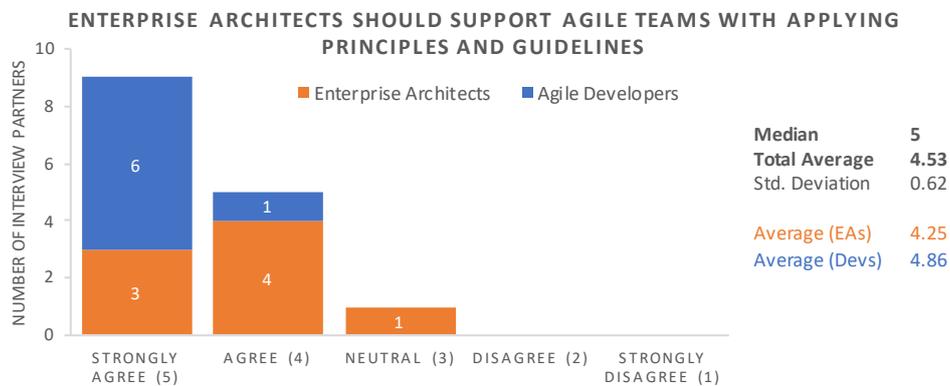


Figure 7.10.: Evaluation results for question 11

also seen as important regarding the knowledge and experience transfer from architects to developers (AD1, AD4). AD2 adds that this could also help architects to get a better idea whether guidelines are actually used. The regular presence of enterprise architects is very important for acceptance and can also help agile teams to overcome their inhibitions to actively approach enterprise architects, thereby leading to agile teams communicating more openly with enterprise architects (AD7). The enterprise architects have a similar point of view and argue that enterprise architects might have additional insight and knowledge that can be helpful for applying a guideline (EA4). The support is also crucial because it ensures that enterprise architects are closer to the teams and gain a deeper understanding of the actual implementation (EA4). The close work with teams also further increases acceptance and provides concrete direct feedback to the enterprise architects (EA2). EA3 raises the question of how exactly the support should look like, e. g. with classical documentation, pair programming or other methods. EA5 stresses that in the approach, the community could also help itself to implement the guidelines. Scaling the support of enterprise architects can be difficult due to capacity constraints (EA8). EA6 takes a similar view and suggests that teams act as self-responsible as possible in applying the principles and guidelines, but they should also be able to request support whenever necessary (EA6).

Question 12: Teams should be self-responsible for managing their compliance with guidelines and not controlled by team-external stakeholders (e. g. enterprise architects) (step 6).

The results show that the self-responsible management of compliance by teams is the most controversially discussed statement of the evaluation questionnaire, with a slight overall disagreement. A positive aspect mentioned is that there is a much better

7. Evaluation

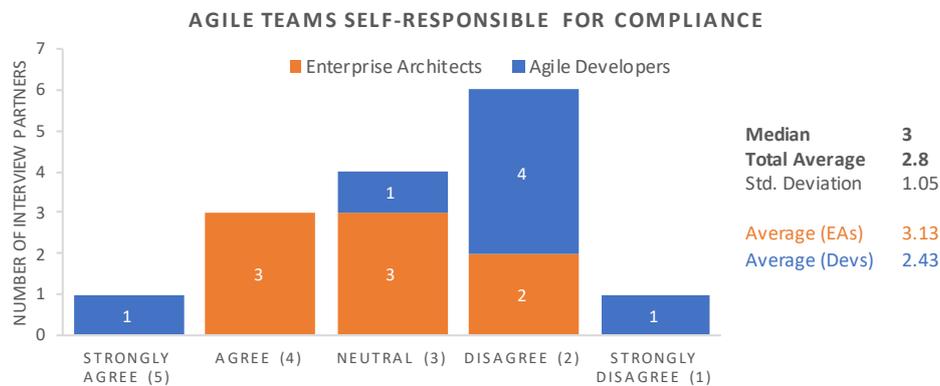


Figure 7.11.: Evaluation results for question 12

chance to shape the sense of responsibility of the teams, so that they more often ask themselves the question "what is the right thing to do?" (AD1). EA6 takes a similar perspective and states that "if you are responsible for something, you take better care of it". If teams are not responsible for managing the compliance themselves, they may also feel that they only need to comply with certain guidelines if someone explicitly notifies the team (EA7). However, most participants state that compliance with guidelines should be controlled. It is great if teams are self-reflective and they should also be granted a leap of faith so that they can decide for themselves (AD7). Nevertheless, at some point, compliance should be verified by an external stakeholder (AD7). If there is no dedicated regulatory body, then perhaps not everything will be implemented (AD3). Teams could do it themselves, but it would be better to monitor and control externally, also to give more time and space to the teams for actual development (AD5). It is highly dependent on the motivation to comply with the guidelines, but if teams participate in the guideline creation, they should also not have a problem with being controlled at this point (AD6). AD4 is in line with this perspective by stating that if the community including developers voted for a specific architecture, everyone should commit themselves to it and it should be controlled accordingly. The control does not necessarily has to be done by an external stakeholder, but at least the developers should control each other (AD4). EA2 states that there should be a way to guarantee compliance to certain "mandatory" guidelines, e. g. the ones that are based on legal requirements. EA5 adds that there is always the need for control in regulated environments. As long as the team does not have the ultimate, full liability over their actions, a complete self-responsible compliance management does not work (EA5). EA8 has the opinion that the full self-responsibility for the compliance of guidelines would be a desirable target state, but it requires a very high level of maturity on the part of

the agile teams and could also be exploited. Governance by external stakeholders can achieve more fairness by having the possibility to intervene and initiate consequences, if necessary (EA8).

7.3. Evaluation of the prototype

7.3.1. Assessment of the main goals of core features

The participants of the evaluation were asked to rate each of the following statements of the questionnaire twice. Once, for rating the situation without the support of the presented tool, and once, with the support of the tool. This is done to obtain reference values of the current situation and be able to compare those with the ones acquired from the assessment of the situation with tool support. Thereby, it is possible to assess whether the tool support fulfills its main goals by improving the current situation significantly. Contrary to the previous evaluation section, in which the participants were asked to choose their answers as universally as possible, driven mainly from their general experience in their role and less from case organization specific experience, the current situation now relates to the situation at the case organization presented in Chapter 4. Figure 7.12 shows the results of the evaluation of the tool support based on the questions that are briefly described below. The results indicate that the situation is already clearly improved with the help of the prototypical web application. Nevertheless, together with additional qualitative feedback by the users, it also shows there is still potential for further improvements and extensions of the web application.

Question 1: Agile teams and enterprise architects have a good overview on which guidelines agile teams must or should adhere to.

The question relates mainly to the team-specific guideline overview feature as presented in Subsection 6.3.1 as well as the guideline selection feature of the application and its goal to provide a better overview of the guidelines.

Question 2: It is easy for agile teams and enterprise architects to give feedback on guidelines.

This question relates to the feedback feature presented in the next steps in Section 6.7 and shown in Figure 6.11. The feature was prototypically implemented for the purpose of the demonstration and evaluation, but the data is not further processed and persisted yet.

Question 3: It is a lot of effort to identify which guidelines a specific agile team is fulfilling and which ones it is not fulfilling.

7. Evaluation

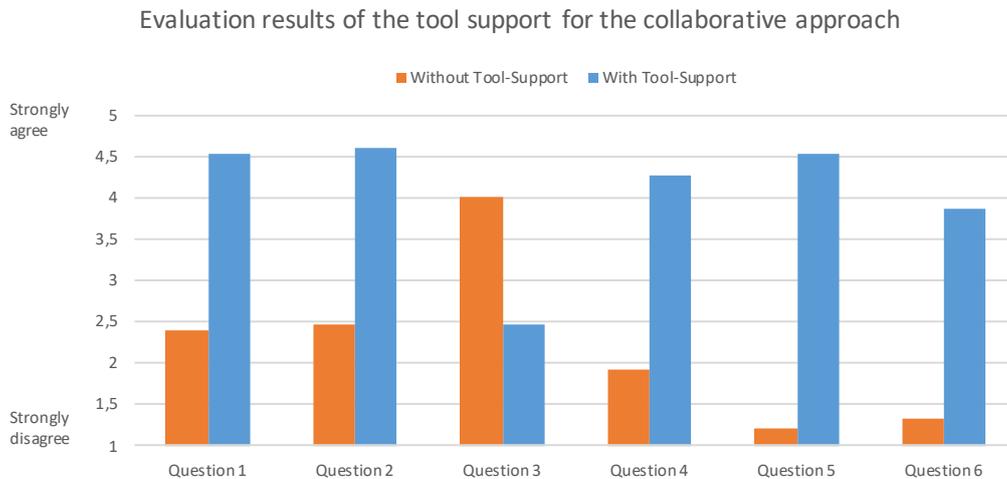


Figure 7.12.: Evaluation results of the tool support

This question relates to the team-specific overview of guidelines, including fulfillment indicators for each of the guidelines that are used by a team, which is represented by green and red bars, as described in Subsection 6.3.1.

Question 4: It is easy for agile teams to look through existing guidelines and select the guidelines that fit their use cases.

This question revolves mainly around the select existing guideline feature and the mapping of guidelines to project characteristics when creating a new team.

Question 5: It is simple to get an overview of which agile teams are compliant to which guidelines.

The question addresses the possibility to select between multiple teams to receive insights and transparency into the guideline compliance and usage status as well as the statistics screen summarizing multiple teams and their compliance with guidelines, as described in Section 6.3.5.

Question 6: It is clear why an agile team decided against using a specific guideline.

This question relates to the possibility for users to enter a team specific solution and rationale for their guideline, describing how they deviate from the standard that is originally intended by the guideline, as mentioned in Section 6.3.2.

7.3.2. Usability assessment based on the System Usability Scale

For the evaluation of our prototypical implementation, we use the "System Usability Scale" (SUS) by Brooke [18] to collect relevant subjective ratings of the usability by our participants who all belong to roles that are part of the potential user group of the application. The SUS is a simple set of ten questions to assess the usability of a system based on a Likert scale, giving the respondents the possibility to rate their agreement or disagreement on regarding ten usability questions on a 5-point scale. The results from an empirical evaluation, based on almost ten years of SUS data collected on multiple products in various phases of the development lifecycle, shows that the "SUS is a highly robust and versatile tool for usability professionals" [12]. During the application of the SUS, it is important to encourage respondents to record their immediate response to each item, instead of thinking about each question for a longer time [18]. Furthermore, respondents should check all items. If respondents are in doubt whether it is possible for them to answer a particular item, the center point of the scale can be marked to indicate a neutral or conflicting stance on the question. The SUS results in one single number on a range from 0 (negative) to 100 (positive), "representing a composite measure of the overall usability of the system being studied" [18]. Brooke also notes that scores for individual items do not provide meaningful insights on their own. Brooke describes the calculation of a system usability score as follows: *"To calculate the SUS score, first sum the score contributions from each item. Each item's score contribution will range from 0 to 4. For items 1, 3, 5, 7, and 9 the score contribution is the scale position minus 1. For items 2, 4, 6, 8, and 10, the contribution is 5 minus the scale position. Multiply the sum of the scores by 2.5 to obtain the overall value of SU"* [18].

The different calculations for the questions are caused by alternating positive and negative items. For positive items, a high usability requires to answer "strongly agree", whereas, for negative ones, a high perceived usability requires the respondent to answer with "strongly disagree". Brooke argues that this was done "in order to prevent response biases caused by respondents not having to think about each statement; by alternating positive and negative items, the respondent has to read each statement and make an effort to think whether they agree or disagree with it [18]".

While such a scoring from 0 to 100 is seen as intuitive and allows for relative judgments [11], the original methodology does not provide a way on how to translate the numeric score into a qualitative judgment of usability. However, Bangor et al. [11] demonstrate such a method and provide the interpretation of a SUS score shown in Figure 7.13, based on the insights of their analysis of almost one thousand SUS surveys that they complemented with a seven-point Likert scale based on evaluative adjectives. Their research shows that their newly introduced Likert scale scores based on seven adjectives reaching from *worst imaginable* to *best imaginable* "correlate extremely well

7. Evaluation

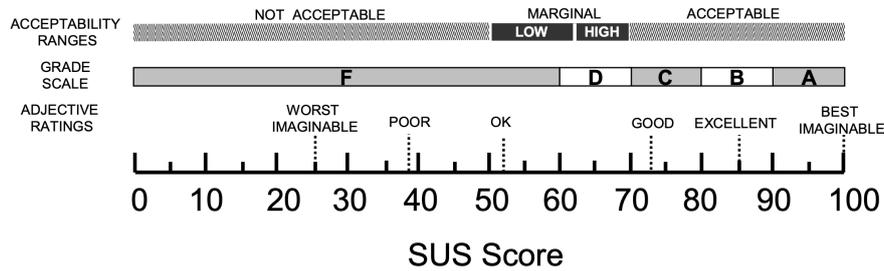


Figure 7.13.: A comparison of the adjective ratings, acceptability scores, and school grading scales, in relation to the average SUS score by Bangor [11]

with the SUS scores ($r=0.822$)" [11].

Table 7.2 shows the results of the usability evaluation. The average of all fifteen usability evaluations result in an **81.33** system usability score, which is in the acceptable region, scoring a "B" in the grading scale and a rating between *good* and *excellent* on the adjective rating scale. Overall, this is a satisfactory result, nevertheless, together with the qualitative feedback and comments of the participants, it also shows room for potential improvements and further development. The obtained system usability score result can be used to compare the results of the current prototype iteration with future development iterations to check whether UI/UX and general usability improvements have the desired effect and are reflected in a higher usability score.

No.	System Usability Score
1	77.5
2	80
3	57.5
4	87.5
5	70
6	87.5
7	90
8	95
9	85
10	80
11	77.5
12	67.5
13	77.5
14	92.5
15	95
Average	81.33

Table 7.2.: System Usability Score results

7.4. Summary of the evaluation results

Figure 7.14 shows an overview of all questions and the resulting level of agreement or disagreement by the two stakeholder groups regarding the first solution artifact of this thesis, the collaborative approach.

With regards to the second solution artifact, the tool support, Figure 7.12 shows that the main goals of the application are being fulfilled by visualizing the comparison of the results between the current situation without tool support and the situation with tool support. Participants rate the overview of an existing guideline, the transparency of the fulfillment status of guidelines (both within a team and across several teams) and the selection of guidelines for a certain use case as considerably improved with the tool-support compared to without the tool-support. The evaluation results also reveal another interesting finding. Whereas some interviewed experts assess the current situation as decent and especially mention the existing documentation on guidelines in the wiki, others point out the lack of useful information. This shows that the wiki can be an appropriate tool, but it also reflects the challenges identified during the case study, which are the lack of a single source of truth and awareness.

7. Evaluation

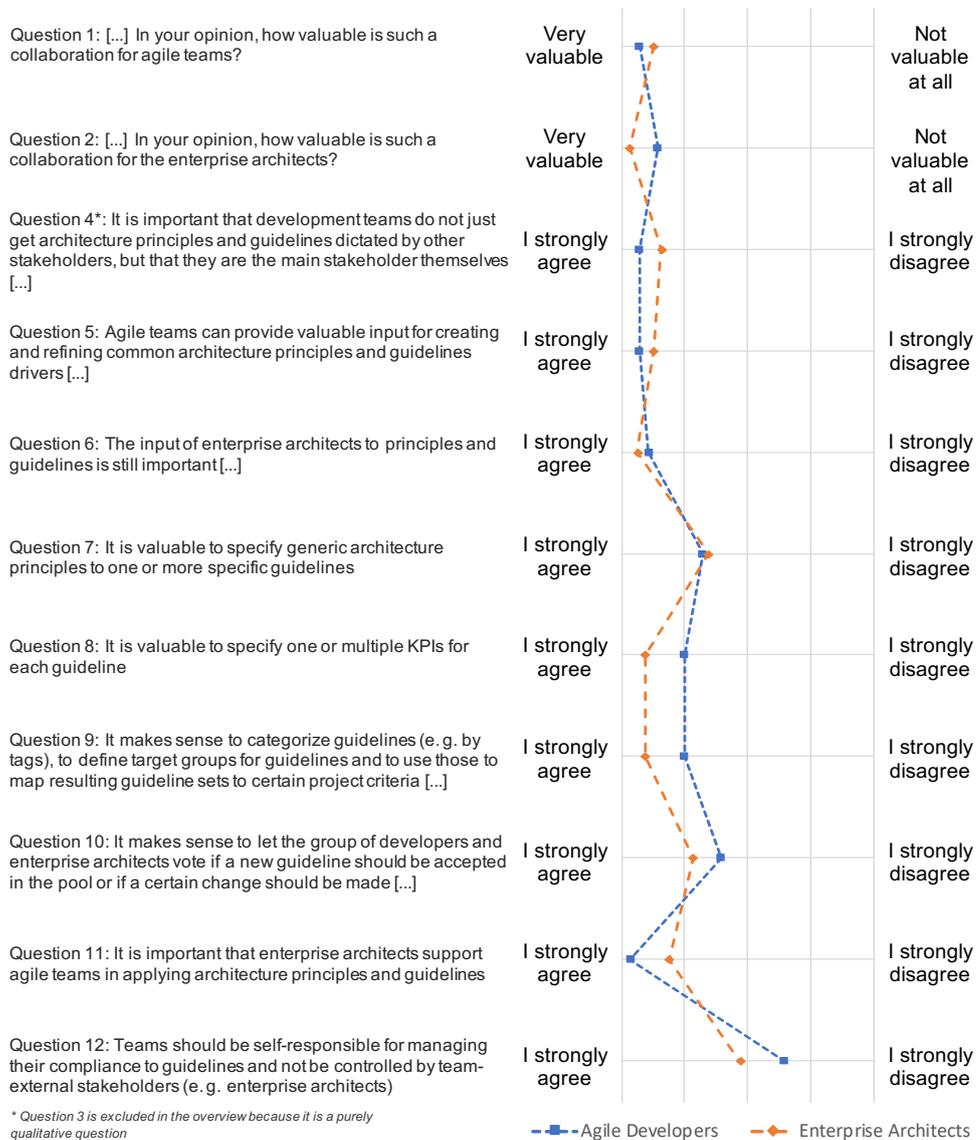


Figure 7.14.: Overview of the evaluation results of the collaborative approach

8. Discussion

This chapter summarizes key findings and provides a critical reflection that discusses the potential limitations of this master's thesis.

8.1. Key findings

In the following, this section describes and summarizes the key findings of this master's thesis.

Strong need for closer collaboration between enterprise architects and agile teams

There is a strong demand for closer collaboration between enterprise architects and agile teams, both in the research described in current literature as well as in practice. The main reasons for this need are to facilitate a better understanding and acceptance between the two stakeholder groups, and to incorporate strategical and operational perspectives and experiences into governance efforts, thereby improving the quality of governance and enterprise architecture artifacts, such as principles and guidelines, and increasing compliance with these governance efforts by agile teams. The need for a closer collaboration was also evaluated during the expert interviews, where there was a strong consensus for the need of the collaborative aspects presented in the solution artifacts of the thesis.

High importance of incorporating the bottom-up perspective into governance and enterprise architecture processes

Incorporating the bottom-up perspective into governance and enterprise architecture efforts can improve the acceptance of the affected groups who actually have to implement the standards prescribed by those governance efforts. Furthermore, the operational perspective can also improve the quality of those efforts by identifying and addressing issues that arise on an operational level, monitor and evaluate the actual impact and effect that existing governance has on an implementation level. Including opera-

tional stakeholders can also improve the relevance and applicability of principles and guidelines.

A community is suited to combine top-down and bottom-up perspectives by jointly creating and managing architecture principles and guidelines

One of our main contributions with this thesis is the presented collaborative approach which provides a basis for establishing a community of relevant stakeholders with insight into some of the key activities that should be conducted in a collaborative way within and around such a community. Those activities include the determination of drivers, deriving principles and guidelines from drivers, specifying and classifying principles and guidelines, voting and accepting and applying them as well as managing their compliance and handling change. The insights from the comprehensive evaluation in fifteen interviews with agile developers and enterprise architects, further detailing the advantages and challenges or potential problems of the approach, should be taken into account during the implementation.

Changing role of enterprise architects in agile environments

The research also indicates the changing role of enterprise architects in agile environments. Enterprise architects are less able to rely on control and authority when working with agile teams, but have to put more focus on enabling and supporting them. They have to ensure that they can provide actual value to the relevant stakeholders. This requires more close collaboration, not only in form of regular community meetings, but also during everyday work, thereby getting insight into problems and challenges agile teams face on day to day basis. Nevertheless, despite the changing role of the enterprise architects in the agile environment, enterprise architects still remain important. For example, they can provide valuable input for agile teams based on their experience, their knowledge and oversight on architectural topics and their insight into multiple teams and business aspects of the organization.

Value of a collaborative and liberal IT-governance, suitable for agile environments

The collaborative form of IT-governance presented in the thesis aims to increase the acceptance of governance efforts by key stakeholders because of their involvement and right to have a say within the governance processes. This, in turn, may increase compliance. Also, the quality of governance artifacts such as principles and guidelines can be increased by achieving a higher relevance and applicability during actual

implementation. Especially in large-scale agile development environments, governance and standardization are still important for a common direction and commitment of all relevant stakeholders, required for achieving organizational agility.

Agile teams do not insist on full autonomy, as long as they have a say in rule-making

As long as they have a say in rule-making, agile developers do not necessarily insist on full autonomy or the right to manage the compliance with existing principles and guidelines all by themselves. Quite the contrary, they believe that it can still be important to have some external stakeholders evaluating the compliance, also saving time and enabling teams to focus on their core task, the development of their software product. Nevertheless, having the ability to participate in the creation of principles and guidelines is essential for agile teams. If this condition is met, then a more strict governance process regarding compliance can help to ensure the actual usage, resulting in a better common direction of all teams. This can be especially helpful in an early state in when collaborative and agile processes have a low maturity in an organization. The full autonomy of agile teams could be a target state in more mature organizations with teams that are already more experienced in self-responsible decision making, not only solely focusing on the local optimum but also taking the global optimum into account.

A web application can be valuable for supporting the collaborative use of principles and guidelines

A web application, with features as presented in the thesis, can provide valuable support for collaboratively using architecture principles and guidelines. So far, the proposed web application mainly aims to increase the transparency and accessibility of guidelines, by supporting to reach a better overview of existing guidelines, selecting relevant guidelines for a specific use case and managing the compliance with guidelines, both within a single team, but also over multiple teams. The prototypical web application also provides the basis for future enhancements, such as the automated testing of guidelines.

8.2. Limitations

This section discusses limitations and threats to the validity of the thesis. First of all, a limitation is the limited time frame of the thesis that is available for conducting the research, which does not allow for a longer evaluation and actual usage of the solution

artifacts over the course of multiple months. Another limitation is the potentially limited generalizability because of the execution of a single case study, focusing on one organization. We aimed to counteract this limitation by including existing literature and related work into the design of our solution artifacts as well as presenting the results of our research to other partner companies of the research chair during a workshop. This provides valuable input and confirmation that the identified challenges and solution artifacts are relevant in other organizations as well.

In regards to counteract the threats of validity concerning the case study part of our research, we addressed the potential threats listed by Runeson and Höst [102] as follows. To address the construct validity, which can be threatened for example when the researcher and interviewed persons have a different understanding of discussed constructs, we interviewed multiple stakeholders with different backgrounds and also emphasize the importance to ask follow up questions in case the nature of a construct is unclear. The external validity deals with how far it is possible to generalize the findings and "to what extent the findings are of interest to other people outside the investigated case" [102]. As stated before, we began to present and discuss the findings with stakeholders outside of the case study company as well, as part of a workshop with other industry partners of the research chair. Nevertheless, regarding the reliability of the research, repeating the research in a different organization might end in different results because in differences of the organizations' culture, structure and maturity regarding agile methods and mindset. Another key limitation in the scope of this thesis regarding the mentioned threats to validity is the diversity and broad nature of possible architecture principles and guidelines, which can be highly specific to certain organizations. Because of this reason, our research focused less on what specific principles and guidelines should be used, and more on the way how principles and guidelines can be used in an organization, independent from the specific nature or category of the applied principles and guidelines. Nonetheless, since principles and guidelines can take on many forms and manifestations, it could be possible that the presented approach and tool support works better for certain types or areas of principles and guidelines.

9. Conclusion and future work

The final chapter of this thesis provides a summary of the thesis based on the research objectives and presents an outlook for further work.

9.1. Summary

The following section summarizes the answers to the research questions presented in Section 1.2.

Research question 1: How does governance and enterprise architecture fit in modern large-scale agile and lean development environments?

Governance and enterprise architecture management is important to ensure a common direction and an alignment of ongoing efforts into the direction of the target vision. This is especially important in large organizations and large-scale agile software development. Nevertheless, the traditional top-down governance approach relying mainly on authority and control is no longer contemporary and does not fit very well with agile and lean principles and values. On the other side, full autonomy of teams would harm the ability of enterprises to achieve organizational agility which requires a high amount of standardization. Instead, more right of co-determination has to be given to the teams, so that they can have a say in governance, facilitating both a higher acceptance as well as quality, relevance, and applicability of governance endeavors. Therefore, the bottom-up perspective of agile teams has to be incorporated into existing processes, turning top-down governance into a more collaborative form of governance.

Research question 2: How can agile teams and enterprise architects collaboratively establish and manage architecture principles and guidelines in large-scale agile software development?

Establishing a community of enterprise architects, agile developers and other interested and relevant stakeholders can enable a close collaboration on principles and guidelines. Taking the bottom-up perspective of teams into account facilitates the consideration of valuable input, including issues that arise on an implementation level, implementation

experience, technical innovation, and team-specific agreements. This contributes to a higher relevance and applicability of principles and guidelines. The community also allows much faster feedback cycles and refinements of existing guidance. The results of governance efforts get much clearer and transparent, since the stakeholders who are responsible for implementing the guidance are now involved in all steps of the process. The results of the semi-structured expert interviews with fourteen participants provide further insight into the value and challenges of the presented approach.

Research question 3: How can the collaborative approach for establishing architecture principles and guidelines be supported and further enhanced by a software implementation?

A web application can provide a suitable form of a software implementation to support the collaborative process. By providing an overview of existing guidelines and their fulfillment for different teams, as well as the possibility to filter and select from existing guidelines, can support agile teams to apply principles and guidelines in their work. Furthermore, creating more transparency and a single source of truth with a web application can support the creation, classification and compliance management of principles and guidelines. It also creates a valuable basis for enabling the partial automation of checking the compliance for principles and guidelines by the ability to connect to various data sources and integrate into existent continuous deployment and delivery pipelines in the future.

9.2. Future work

Due to the limited time frame of this master's thesis, the results of the evaluation could not be implemented in further development iterations. Including the valuable feedback from the conducted expert interviews can be one of the crucial next steps for further work. Besides improving existing traits of the approach and tool-support, the tool-support could also be extended to implement the remaining aspects of the approach which are not yet covered by the current implementation, e. g. managing drivers or the ability to specify and manage KPIs for guidelines.

Another aspect for future work is the generalization of the results. Firstly, applying and evaluating the approach and tool-support in other companies could provide helpful insights on how to further refine and adjust the approach and the demonstrated web application. Secondly, our collaborative approach for establishing guidelines is not inherently limited to principles and guidelines. On the contrary, principles and guidelines are only one possible artifact that can be established collaboratively to achieve agile IT-governance. It is conceivable that other aspects of governance and

enterprise architecture management could also be developed collaboratively, e. g. a target architecture or system landscape, architectural models or more strategical goals and values. In general, more research could be done on whether the transition to more collaborative decision making based on agile and lean values and principles is desirable in organizations and how such an approach can be implemented.

To demonstrate and proof the actual long-term value and applicability of the collaborative approach and tool-support, one of the possible next steps could be a field study obtaining data on the long-term usage, providing more measurements and insights into the applicability and value contribution of the solution artifacts over a longer period of time.

A. Appendix

A.1. Evaluation interviews

1. General questions

- a) Which of the below roles are applicable for you?
- b) For how many years have you been active in the field of enterprise architecture and/or agile software development?

2. Collaborative approach for establishing architecture principles and guidelines

- a) The approach focuses on creating a community of developers and solution architects from different agile teams as well as enterprise architects, meeting regularly, discussing and deciding on architecture principles and guidelines together.
 - i. In your opinion, how valuable is such a collaboration for agile teams?
 - ii. In your opinion, how valuable is such a collaboration for enterprise architects?
- b) What challenges do you see in the actual implementation of such a community? In your opinion, which problems need to be considered and solved in order to make the collaborative approach successful?
- c) It is important that development teams do not just get architecture principles and guidelines dictated by other stakeholders, but that they are the main stakeholder themselves in managing architecture principles and guidelines and are involved in the whole approach.
- d) Agile teams can provide valuable input for creating and refining common architecture principles and guidelines drivers (especially based on their implementation experience, the issues they face on a daily basis, team-specific agreements, and their technical knowledge and insights in innovative technologies) (step 1).
- e) The input of enterprise architects to principles and guidelines is still important because of their insight into multiple teams, knowledge of business

goals, strong network and their resulting ability to align multiple teams, business and IT (overall).

- f) The approach proposes to derive architecture principles from drivers and then further specify those – often more generic – principles into one or multiple, more specific guidelines that give insight on how to actually implement the principle. It proposes to define one or more KPIs for each guideline to be able to measure the level of compliance with a guideline (step 2-3).
 - i. It is valuable to specify generic architecture principles to one or more specific guidelines.
 - ii. It is valuable to specify one or multiple KPIs for each guideline.
- g) It makes sense to categorize guidelines (e. g. by tags), to define target groups for guidelines and to use those tags and target guidelines later to map those resulting sets of guidelines to certain project criteria (because not all guidelines are applicable for all types of projects, but there are certain patterns of important guidelines for similar projects) (step 3).
- h) It makes sense to let the group of developers and (enterprise) architects vote if a new guideline should be accepted into the pool or if a certain change should be made (as long as the new or changed guideline is not based on a legal requirement), instead of solely relying on a senior architect or manager to make that decision alone (step 4).
- i) It is important that enterprise architects support agile teams in applying architecture principles and guidelines (step 5).
- j) Teams should be self-responsible for managing their compliance to guidelines and not controlled by team-external stakeholders (e. g. enterprise architects) (step 6).
- k) Do you have any comments or open points?

3. Tool-Support

- a) Agile teams and enterprise architects have a good overview on which guidelines agile teams must or should adhere to.
- b) It is easy for agile teams and enterprise architects to give feedback on guidelines.
- c) It is a lot of effort to identify which guidelines a specific agile team is fulfilling and which ones it is not fulfilling.

- d) It is easy for agile teams to look through existing guidelines and select the guidelines that fit to their use cases.
- e) It is simple to get an overview of which agile teams are compliant to which guidelines.
- f) It is clear why an agile team decided against using a specific guideline.
- g) General usability of the application
 - i. I think that I would like to use this system frequently
 - ii. I found the system unnecessarily complex
 - iii. I thought the system was easy to use
 - iv. I think that I would like to use this system frequently
 - v. I think that I would need the support of a technical person to be able to use this system
 - vi. I found the various functions in this system were well integrated
 - vii. I thought there was too much inconsistency in this system
 - viii. I would imagine that most people would learn to use this system
 - ix. I found the system very cumbersome to use
 - x. I felt very confident using the system
 - xi. I need to learn a lot of things before I could get going with this system

A.2. Semi-structured case study interviews

A.2.1. General information

1. Section: Overview of the transformation

- a) What are the reasons for the transformation?
- b) What are the goals of the transformation?
- c) When did the transformation begin?
- d) How did the transformation take place?
- e) What success have you achieved so far?

2. Section: Agile methods and practices

- a) Which agile and large scale agile methods / frameworks are used in your organization?

3. Section: Challenges and solutions

- a) What are the greatest challenges of the transformation?
- b) How did you try to solve the challenges?

4. Section: Plans for the future

- a) What are the next steps?
- b) What are possible stumbling blocks?
- c) On a scale from 1 "no enablement at all" to 10 "very strong enablement", how strong does the EAM enable the scaling of agile practices?

A.2.2. Architecture principles

1. Section: Background questions

- a) Which role description applies to you?
- b) For how many years have you been working in the field of agile software development or EAM?
- c) How is your EAM organization structured? Which architectural roles do you have and how are they assigned to the agile teams?

2. Section: Drivers and goals

- a) Which driving forces are significantly responsible for creating architectural principles?
- b) What are your goals in defining architectural principles?

3. Section: Specification and classification

- a) Which architectural principles do you use in an agile environment? Which architectural principles from the attached Excel list do you use? (Please specify them in the Excel list)
- b) Are there architectural principles that are unsuitable for the agile environment or which are not applied? If so, why?
- c) Who is responsible for the creation and specification of the architectural principles?
- d) Is there a guideline for creating architectural principles?
- e) (If yes): With which specifications?
- f) (If yes): Who created this guideline and is responsible for its maintenance?

4. Section: Application and compliance

- a) When do new architectural principles become valid?
- b) What measures exist for the implementation of the newly created architectural principles?
- c) In which way is compliance with the architectural principles checked and ensured?
- d) Which tools are used to test compliance with architectural principles?
- e) What problems do you identify when introducing or implementing architectural principles?

5. Section: Discussion

- a) Generally speaking, what do you expect from the cooperation with TUM?
- b) If necessary, can we contact you again in the context of the case study? If yes, please enter your name and email address. Naturally, we will not make this data available to third parties.
- c) Are there any comments or open questions?

A.2.3. Architecture boards

1. Section: Background Information

- a) Which role description applies to you?
- b) For how many years have you been working in the field of agile software development or EAM?

2. Section: Architectural Decisions

- a) Name typical examples of architectural decisions in an agile environment.
- b) How do you categorize architectural decisions?
- c) How and where are architectural decisions documented?
- d) At what level and by which role is an architectural decision made?

3. Section: Architecture Boards

- a) What forms of architectural boards do you have in your company? Please specify the existing forms in the attached Excel list.

4. Section: discussion

- a) Generally speaking, what do you expect from the cooperation with TUM?
- b) If necessary, can we contact you again in the context of the case study? If yes, please enter your name and email address. Naturally, we will not make this data available to third parties.
- c) Are there any comments or open questions?

A.2.4. Role of the enterprise architect

1. Section: Background information

- a) Which role description applies to you?

- b) For how many years have you been working in the field of agile software development or EAM?

2. Section: Responsibility

- a) What responsibilities do Enterprise Architects have in your company?
- b) How did the responsibilities of Enterprise Architects change in an agile environment?
- c) What distinguishes your Enterprise Architects and how do they differ from the classic role of Enterprise Architects in other companies?
- d) How did the working methodology of the Enterprise Architect change in an agile environment?
- e) How did the environment of the Enterprise Architect change in an agile environment?

3. Section: Collaboration

- a) How should Enterprise Architects support agile programs and agile transformation?
- b) How is the role of the architect practiced in your company?
- c) What are the characteristics of the role selected in question 3.2?
- d) What are the advantages and disadvantages of the role mentioned in question 3.2?
- e) In your opinion, which of the roles listed in question 3.2 should be lived more strongly in the future?
- f) Should Enterprise Architects be part of the agile teams?
- g) To what extent should Enterprise Architects be involved in the agile programs?

4. Section: Artifacts

- a) What artifacts are provided by the Enterprise Architects to the stakeholders in an agile environment?
- b) What artifacts should be provided by Enterprise Architects to stakeholders in the future?

5. Section: Problems and outlook

- a) What problems do Enterprise Architects encounter in an agile environment?
- b) In your opinion, how should these problems be addressed?

- c) What recommendations would you give Enterprise Architects in an agile environment?
- d) On which topics should Enterprise Architects focus in the future?

6. Section: Discussion

- a) Generally speaking, what do you expect from the cooperation with TUM?
- b) If necessary, can we contact you again in the context of the case study? If yes, please enter your name and email address. Naturally, we will not make this data available to third parties.
- c) Are there any comments or open questions?

A.2.5. Value contribution of enterprise architects

1. Section: Background information

- a) Which role description applies to you?
- b) For how many years have you been working in the field of agile software development or EAM?

2. Section: Enabling

- a) What information do Enterprise Architects need from agile teams to provide appropriate architectural models?
- b) Which architectural models are provided to the agile teams by the Enterprise Architects?
- c) What do agile teams expect from Enterprise Architects' architectural models?
- d) Are the expectations regarding the architectural models fulfilled? (On a scale from 1 "not fulfilled" to 10 "more than fulfilled")
- e) Please give reasons for your answer:

3. Section: Cooperation

- a) Is it difficult to find an Enterprise Architect as a contact person?
- b) How do you rate the availability of an Enterprise Architect? (On a scale from 1 "not available at all" to 10 "always available")
- c) How do Enterprise Architects communicate with agile teams?
- d) What do agile teams expect from the communication with Enterprise Architects?

- e) Are expectations in terms of communication fulfilled? (On a scale from 1 "not fulfilled" to 10 "more than fulfilled")
- f) Please give reasons for your answer:
- g) To what extent are agile teams involved in the architecture processes that are relevant to them?
- h) To what extent should agile teams be involved in the architecture processes that are relevant to them?
- i) Are the expectations in terms of the integration into architectural processes fulfilled? (On a scale from 1 "not fulfilled" to 10 "more than fulfilled")
- j) Please give reasons for your answer:
- k) How do Enterprise Architects support agile teams?
- l) What kind of support do agile teams expect from Enterprise Architects?
- m) Are the expectations in terms of support fulfilled? (On a scale from 1 "not fulfilled" to 10 "more than fulfilled")
- n) Please give reasons for your answer:
- o) In what form and frequency can the agile teams give feedback to Enterprise Architects?
- p) In what form should feedback from agile teams be given to Enterprise Architects?
- q) Are the expectations in terms of feedback fulfilled? (On a scale from 1 "not fulfilled" to 10 "more than fulfilled")
- r) Please give reasons for your answer:
- s) On a scale from 1 "not fulfilled" to 10 "more than fulfilled", how likely is it that you would recommend the EA? (To an agile team that has developed without the support of an Enterprise Architect)

4. Section: Review

- a) How does feedback from agile teams affect the EAM?
- b) Which criteria are currently used to assess the value contribution of the EAM and which metrics are used to measure it?
- c) How could the value contribution be measured in the future?

5. Section: Governance

- a) To what extent do agile teams want to be controlled by the Enterprise Architects?
- b) Which requirements can help?
- c) Which specifications would not be helpful or restrictive?

6. Section: Discussion

- a) Generally speaking, what do you expect from the cooperation with TUM?
- b) If necessary, can we contact you again in the context of the case study? If yes, please enter your name and email address. Naturally, we will not make this data available to third parties.
- c) Are there any comments or open questions?

Bibliography

- [1] P. Abrahamsson, M. A. Babar, and P. Kruchten. "Agility and architecture: Can they coexist?" In: *IEEE Software* (2010). ISSN: 07407459. DOI: 10.1109/MS.2010.36.
- [2] F. Ahlemann, E. Stettiner, M. Messerschmidt, and C. Legner. *Strategic Enterprise Architecture Management: Challenges, Best Practices, and Future Developments*. 2012. ISBN: 3642242227. DOI: 10.1007/978-3-642-24223-6.
- [3] S. Aier, C. Riege, and R. Winter. "Unternehmensarchitektur – Literaturüberblick und Stand der Praxis". In: *WIRTSCHAFTSINFORMATIK* (2008). ISSN: 0937-6429. DOI: 10.1365/s11576-008-0062-9.
- [4] M. Alqudah and R. Razali. "A Review of Scaling Agile Methods in Large Software Development". In: *International Journal on Advanced Science, Engineering and Information Technology* (2016). ISSN: 2460-6952. DOI: 10.18517/ijaseit.6.6.1374.
- [5] S. Ambler. *The Disciplined Agile Framework 2.0*. URL: <http://www.disciplinedagiledelivery.com>.
- [6] S. Ambler. *The Disciplined Agile Framework 2.0: Agility at scale and enterprise architecture*. 2016. URL: <http://www.disciplinedagiledelivery.com/agility-at-scale/enterprise-architecture/>.
- [7] S. W. Ambler. "Scaling agile software development through lean governance". In: *Proceedings of the 2009 ICSE Workshop on Software Development Governance, SDG 2009*. 2009. ISBN: 9781424437368. DOI: 10.1109/SDG.2009.5071328.
- [8] S. Ambler and P. Kroll. *Lean development governance: Applying agile and lean principles to the governance of software and systems development*. Tech. rep. IBM, 2007.
- [9] M. A. Babar. "An exploratory study of architectural practices and challenges in using agile software development approaches". In: *2009 Joint Working IEEE/IFIP Conference on Software Architecture European Conference on Software Architecture*. Sept. 2009, pp. 81–90. DOI: 10.1109/WICSA.2009.5290794.
- [10] M. A. Babar, A. W. Brown, and I. Mistrik. *Agile Software Architecture: Aligning Agile Processes and Software Architectures*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013. ISBN: 9780124077720.

- [11] A. Bangor, P. Kortum, and J. Miller. "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale". In: *Journal of Usability Studies* (2009). ISSN: 1931-3357.
- [12] A. Bangor, P. T. Kortum, and J. T. Miller. "An empirical evaluation of the system usability scale". In: *International Journal of Human-Computer Interaction* (2008). ISSN: 10447318. DOI: 10.1080/10447310802205776.
- [13] J. M. Bass. "How product owner teams scale agile methods to large distributed enterprises". In: *Empirical Software Engineering* (2015). ISSN: 15737616. DOI: 10.1007/s10664-014-9322-z.
- [14] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. *Manifesto for Agile Software Development*. 2001. URL: <http://agilemanifesto.org/>.
- [15] S. Bente, U. Bombosch, and S. Langade. *Collaborative Enterprise Architecture*. 2012. ISBN: 9780124159341. DOI: 10.1016/C2011-0-69682-6.
- [16] P. van Bommel, P. M. Buitenhuis, S. J. Proper, and E. H. Hoppenbrouwers. "Architecture principles – A regulative perspective on enterprise architecture". In: *Enterprise modelling and information systems architectures : concepts and applications ; proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures, St. Goar, Germany, October 8 - 9. 2007*. ISBN: 9789461911780. DOI: 10.1063/1.3033202. arXiv: 0512156v1 [cond-mat].
- [17] S. Bradner. *RFC 2119 - Key words for use in RFCs to Indicate Requirement Levels Status*. 1997. DOI: <http://www.ietf.org/rfc/rfc2119.txt>.
- [18] J. Brooke. "SUS - A quick and dirty usability scale". In: *Usability Evaluation in Industry*. 1996. ISBN: 0748404600.
- [19] M. Brosius, S. Aier, K. Haki, and R. Winter. "Enterprise Architecture Assimilation: An Institutional Perspective". In: *Thirty Ninth International Conference on Information Systems (ICIS 2018)*. San Francisco, CA: Association for Information Systems, 2018, pp. 1–16. URL: <https://www.alexandria.unisg.ch/255633/>.
- [20] C. V. Brown. "Examining the Emergence of Hybrid IS Governance Solutions: Evidence from a Single Case Site". In: *Information Systems Research* (1997). ISSN: 10477047. DOI: 10.1287/isre.8.1.69.
- [21] S. Buckl and C. Schweda. *On the State-of-the-Art in Enterprise Architecture Management Literature*. Tech. rep. Chair for Software Engineering of Business Information Systems. Technische Universität München, 2011.

- [22] M. Canat, N. P. Català, A. Jourkovski, S. Petrov, M. Wellme, and R. Lagerström. “Enterprise Architecture and Agile Development - Friends or Foes?” In: *IEEE 22nd International Enterprise Distributed Object Computing Workshop*. 2018. URL: <https://tearseriesdotorg.wordpress.com/tear2018/>.
- [23] T. H. Cheng, S. Jansen, and M. Remmers. “Controlling and monitoring agile software development in three dutch product software companies”. In: *Proceedings of the 2009 ICSE Workshop on Software Development Governance, SDG 2009*. 2009. ISBN: 9781424437368. DOI: 10.1109/SDG.2009.5071334.
- [24] Y.-K. Chou. *Actionable Gamification : beyond points, badges, and leaderboards*. Octalysis Media, 2015. ISBN: 9781511744041.
- [25] S. R. Clegg, T. S. Pitsis, T. Rura-Polley, and M. Marosszeky. *Governmentality Matters: Designing an Alliance Culture of Inter-organizational Collaboration for Managing Projects*. 2002. DOI: 10.1177/0170840602233001.
- [26] A. Cleven, P. Gubler, and K. M. Hüner. “Design alternatives for the evaluation of design science research artifacts”. In: *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology - DESRIST '09*. 2009. ISBN: 9781605584089. DOI: 10.1145/1555619.1555645.
- [27] K. Conboy. “Agility from first principles: Reconstructing the concept of agility in information systems development”. In: *Information Systems Research* (2009). ISSN: 15265536. DOI: 10.1287/isre.1090.0236.
- [28] K. Conboy and B. Fitzgerald. “The views of experts on the current state of agile method tailoring”. In: *IFIP International Federation for Information Processing*. 2007. ISBN: 0387728031.
- [29] K. Dikert, M. Paasivaara, and C. Lassenius. “Challenges and success factors for large-scale agile transformations: A systematic literature review”. In: *Journal of Systems and Software* (2016). ISSN: 01641212. DOI: 10.1016/j.jss.2016.06.013.
- [30] T. Dingsøy, T. E. Fægri, and J. Itkonen. “What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development”. In: *Product-Focused Software Process Improvement* 8892 (2014), pp. 273–276. ISSN: 0302-9743. DOI: 10.1007/978-3-319-13835-0_{_}20.
- [31] T. Dingsøy, N. B. Moe, T. E. Fægri, and E. A. Seim. “Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation”. In: *Empirical Software Engineering* (2018). ISSN: 15737616. DOI: 10.1007/s10664-017-9524-2.

- [32] T. Dingsøy, S. Nerur, V. Balijepally, and N. B. Moe. "A decade of agile methodologies: Towards explaining agile software development". In: *Journal of Systems and Software* (2012). ISSN: 01641212. DOI: 10.1016/j.jss.2012.02.033.
- [33] T. Dreesen and T. Schmid. "Do As You Want Or Do As You Are Told? Control vs. Autonomy in Agile Software Development Teams". In: *Proceedings of the 51st Hawaii International Conference on System Sciences*. 2018. ISBN: 0998133116.
- [34] T. Dybå and T. Dingsøy. *Empirical studies of agile software development: A systematic review*. 2008. DOI: 10.1016/j.infsof.2008.01.006.
- [35] J. Eckstein. "Architecture in Large Scale Agile Development". In: *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*. Ed. by T. Dingsøy, N. B. Moe, R. Tonelli, S. Counsell, C. Gencel, and K. Petersen. Cham: Springer International Publishing, 2014, pp. 21–29. ISBN: 978-3-319-14358-3.
- [36] S. Freudenberg and H. Sharp. "The top 10 burning research questions from practitioners". In: *IEEE Software* (2010). ISSN: 07407459. DOI: 10.1109/MS.2010.129.
- [37] D. Greefhorst and E. Proper. *Architecture Principles The Cornerstones of Enterprise Architecture*. 2011. ISBN: 9783642202780. DOI: 10.1007/978-3-642-20279-7.
- [38] D. Greefhorst and H. Proper. "A Practical Approach to the Formulation and Use of Architecture Principles". In: *2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops*. IEEE, Aug. 2011, pp. 330–339. ISBN: 978-1-4577-0869-5. DOI: 10.1109/EDOCW.2011.18. URL: <http://ieeexplore.ieee.org/document/6037636/>.
- [39] S. H. Spewak and S. C. Hill. *Enterprise Architecture Planning : Developing a Blueprint for Data, Applications and Technology*. Wiley-QED, 1993. ISBN: 0471599859.
- [40] M. K. Haki and C. Legner. "Enterprise Architecture Principles in Research and Practice: Insights from an Exploratory Analysis". In: *Proceedings of the 21st European Conference on Information Systems ECIS*. 2013.
- [41] G. Hamel. "The why, what, and how of management innovation". In: *Harvard Business Review* 84.2 (2006).
- [42] S. Hanschke, J. Ernsting, and H. Kuchen. "Integrating Agile Software Development and Enterprise Architecture Management". In: *48th Hawaii International Conference on System Sciences*. 2015. ISBN: 9781479973675. DOI: 10.1109/HICSS.2015.492.
- [43] M. Hauder, S. Roth, C. Schulz, and F. Matthes. "Agile Enterprise Architecture Management: An Analysis on the Application of Agile Principles". In: *Fourth International Symposium on Business Modeling and Software Design (BMSD)* (2014).

- [44] J. C. Henderson and H. Venkatraman. "Strategic alignment: Leveraging information technology for transforming organizations". In: *IBM Systems Journal* 32.1 (1993), pp. 472–484. ISSN: 0018-8670. DOI: 10.1147/sj.382.0472.
- [45] E. Herranz, R. Colomo-Palacios, A. de Amescua Seco, and M. Yilmaz. "Gamification as a disruptive factor in software process improvement initiatives". In: *Journal of Universal Computer Science* (2014). ISSN: 09486968. DOI: 10.3217/jucs-020-06-0885.
- [46] A. R. Hevner, S. T. March, J. Park, and S. Ram. "Design Science in Information Systems Research". In: *MIS Quarterly* (2004). ISSN: 02767783. DOI: 10.2307/25148625.
- [47] S. Hino. *Inside the mind of Toyota: management principles for enduring growth*. New York: Productivity Press New York, 2006. ISBN: 1563273004.
- [48] R. Hoda, N. Salleh, and J. Grundy. "The Rise and Evolution of Agile Software Development". In: *IEEE Software* (2018), p. 1. ISSN: 0740-7459. DOI: 10.1109/MS.2018.290111318. URL: doi.ieeecomputersociety.org/10.1109/MS.2018.290111318.
- [49] J. Hoogervorst. *Enterprise Governance and Enterprise Engineering*. Springer, 2009. ISBN: 978-3-540-92670-2. DOI: 10.1007/978-3-540-92671-9.
- [50] B. Horlach, T. Böhm, I. Schirmer, and P. Drews. "IT Governance in Scaling Agile Frameworks". In: *Multikonferenz Wirtschaftsinformatik (MKWI) 2018*. Lüneburg, Germany, 2018, pp. 1789–1800.
- [51] J. A. Ingvaldsen and M. Rolfsen. "Autonomous work groups and the challenge of inter-group coordination". In: *Human Relations* (2012). ISSN: 00187267. DOI: 10.1177/0018726712448203.
- [52] ISACA. *COBIT 5 - A Business Framework for the Governance and Management of Enterprise IT*. Rolling Meadows, IL, USA, 2012. ISBN: 978-1-60420-241-0.
- [53] ISO and IEC. *ISO/IEC 38500:2015 International Standard Information technology - Governance of IT for the organization*. Tech. rep. International Organization for Standardization, 2015. URL: <https://www.iso.org/standard/62816.html>.
- [54] ISO, IEC, and IEEE. "ISO/IEC/IEEE 42010:2011 - Systems and software engineering – Architecture description". In: *ISO / IEC / IEEE Standard 42010:2011 (Revision of ISOIEC 42010:2007 and IEEE Std 1471:2000)* (2011). DOI: 10.1109/IEEESTD.2011.6129467.
- [55] D. D. Jacobson. "Revisiting IT governance in the light of institutional theory". In: *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*. 2009. ISBN: 9780769534503. DOI: 10.1109/HICSS.2009.374.

- [56] A. J.H.de O.Luna, P. Kruchten, M. L. d. E.Pedrosa, H. R. Almeida Neto, and H. P. M. Moura. "State of the Art of Agile Governance: A Systematic Review". In: *International Journal of Computer Science and Information Technology* (2014). ISSN: 09754660. DOI: 10.5121/ijcsit.2014.6510.
- [57] K. Julia, S. Kurt, and S. Ulf. "Challenges in Integrating Product-IT into Enterprise Architecture - A case study". In: *Procedia Computer Science*. 2017. DOI: 10.1016/j.procs.2017.11.070.
- [58] J. J. Korhonen, J. Lapalme, D. McDavid, and A. Q. Gill. "Adaptive Enterprise Architecture for the Future: Towards a Reconceptualization of EA". In: *2016 IEEE 18th Conference on Business Informatics (CBI)*. Vol. 01. Aug. 2016, pp. 272–281. DOI: 10.1109/CBI.2016.38.
- [59] J. J. Korhonen, K. Hiekkanen, and J. Lähteenmäki. "EA and IT governance- A systemic approach". In: *European Conference on Leadership, Management and Governance*. 2009.
- [60] P. Kruchten. "Contextualizing agile software development". In: *Journal of software: Evolution and Process*. 2013. DOI: 10.1002/smr.572.
- [61] T. Kude, M. Lazic, A. Heinzl, and A. Neff. "Achieving IT-based synergies through regulation-oriented and consensus-oriented IT governance capabilities". In: *Information Systems Journal* 28 (2018), pp. 765–795. ISSN: 13652575. DOI: 10.1111/isj.12159.
- [62] L. Lagerberg, T. Skude, P. Emanuelsson, K. Sandahl, and D. Stahl. "The impact of agile principles and practices on large-scale software development projects: A multiple-case study of two projects at Ericsson". In: *International Symposium on Empirical Software Engineering and Measurement*. 2013. DOI: 10.1109/ESEM.2013.53.
- [63] M. O. L. Land, E. Proper, M. Waage, J. Cloo, and C. Steghuis. "Enterprise Architecture: Creating Value by Informed Governance". In: *Springer* (2009). ISSN: 1098-6596. DOI: 10.1007/978-3-540-85232-2.
- [64] M. M. Lankhorst and H. A. Proper. "Agile Architecture". In: *Agile Service Development: Combining Adaptive Methods and Flexible Solutions*. Ed. by M. Lankhorst. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 41–57. ISBN: 978-3-642-28188-4. URL: https://doi.org/10.1007/978-3-642-28188-4_3http://link.springer.com/10.1007/978-3-642-28188-4_3.
- [65] M. Lazic, M. Groth, C. Schillinger, and A. Heinzl. "The Impact of IT Governance on Business Performance." In: *AMCIS 2011*. Citeseer. 2011. URL: https://aisel.aisnet.org/amcis2011_submissions/189.

- [66] A. Leclercq-Vandelannoitte and B. Emmanuel. "From sovereign IT governance to liberal IT governmentality? A Foucauldian analogy". In: *European Journal of Information Systems* 27.3 (2018), pp. 326–346. ISSN: 0960-085X. DOI: 10.1080/0960085X.2018.1473932. URL: <https://www.tandfonline.com/doi/full/10.1080/0960085X.2018.1473932>.
- [67] D. Leffingwell. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. 1st. Addison-Wesley Professional, 2011. ISBN: 978-0-321-63584-6.
- [68] M. Leih. "IT Governance and the Sarbanes-Oxley Act". In: *AMCIS 2006 Proceedings*. Vol. 129. 2006, pp. 970–979.
- [69] H. L. H. Lingyu, L. B. L. Bingwu, Y. R. Y. Ruiping, and W. J. W. Jianzhang. "An IT Governance Framework of ERP System Implementation". In: *Computing, Control and Industrial Engineering (CCIE), 2010 International Conference on* (2010). DOI: 10.1109/CCIE.2010.226.
- [70] T. Lumor, E. Chew, and A. Q. Gill. "Exploring the Role of Enterprise Architecture in IS-enabled Ot: An EA Principles Perspective". In: *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW*. 2016. ISBN: 9781467399333. DOI: 10.1109/EDOCW.2016.7584360.
- [71] A. J. H. d. O. Luna, P. Kruchten, E. L. Riccio, and H. P. d. Moura. "Foundations for an Agile Governance Manifesto: a bridge for business agility". In: *13th International Conference on Management of Technology and Information Systems* (2016).
- [72] A. J. H. D. O. Luna, C. P. Costa, and C. A. D. C. Nascimento. "Agile Governance in Information and Communication Technologies : Shifting Paradigms". In: *Journal of Information Systems a Technology Management* (2010). ISSN: 18071775. DOI: 10.4301/S1807-17752010000200004.
- [73] E. Marks. *A Lean Enterprise Governance Manifesto: Improving Business Performance with Event-Driven Governance*. Tech. rep. AgilePath, 2012.
- [74] R. Marques, C. Gonçalo, D. Gonçalves, M. Mira da Silva, and P. Gonçalves. "Improving Scrum Adoption with Gamification". In: *Twenty-fourth Americas Conference on Information Systems*. New Orleans, 2018.
- [75] L. M. Maruping, V. Venkatesh, and R. Agarwal. "A control theory perspective on agile methodology use and changing user requirements". In: *Information Systems Research* (2009). ISSN: 1526-5536. DOI: 10.1287/isre.1090.0238.
- [76] F. Matthes, I. Monahov, S. Alexander, and C. Schulz. *EAM KPI Catalog*. 2011.

Bibliography

- [77] Á. Medinilla. *Agile management: Leadership in an agile environment*. 2012. ISBN: 9783642289095. DOI: 10.1007/978-3-642-28909-5.
- [78] *Merriam-Webster Online Dictionary: "guideline" definition*. URL: <https://www.merriam-webster.com/dictionary/guideline>.
- [79] *Merriam-Webster Online Dictionary: "principle" definition*. URL: <https://www.merriam-webster.com/dictionary/principle>.
- [80] M. Meyer, R. Zarnekow, and L. M. Kolbe. "IT-Governance - Begriff, Status quo und Bedeutung". In: *Wirtschaftsinformatik* (2003). ISSN: 09376429.
- [81] N. B. Moe, D. Šmite, A. Šāblis, A.-L. Börjesson, and P. Andréasson. "Networking in a large-scale distributed agile project". In: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14*. 2014. ISBN: 9781450327749. DOI: 10.1145/2652524.2652584.
- [82] S. Newman. *Building Microservices*. O'Reilly Media, Inc., 2015. ISBN: 9781491950357. DOI: 10.1109/MS.2016.64.
- [83] K. Niemann. *Von der Unternehmensarchitektur zur IT-Governance: Bausteine für ein wirksames IT-Management*. Vieweg+Teubner Verlag, 2005. ISBN: 978-3-528-05856-2.
- [84] E. Niemi and K. S. Soliman. "Enterprise architecture benefits: Perceptions from literature and practice". In: *7th IBIMA Conference Internet & Information Systems in the Digital Age*. 2006. ISBN: 9780975339367.
- [85] R. L. Nord, I. Ozkaya, and P. Kruchten. "Agile in Distress: Architecture to the Rescue". In: *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*. Ed. by T. Dingsøyr, N. B. Moe, R. Tonelli, S. Counsell, C. Gencel, and K. Petersen. Cham: Springer International Publishing, 2014, pp. 43–57. ISBN: 978-3-319-14358-3.
- [86] Object Management Group. *Unified Modeling Language Specification Version 2.5.1*. 2017. URL: <https://www.omg.org/spec/UML/>.
- [87] M. Op't Land and E. Proper. "Impact of Principles on Enterprise Engineering." In: *ECIS*. 2007, pp. 1965–1976.
- [88] K. Peffers, M. Rothenberger, T. Tuunanen, and R. Vaezi. "Design science research evaluation". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2012. ISBN: 9783642298622.
- [89] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* (2007). ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302.

- [90] R. Pereira and M. M. da Silva. "A Literature Review: Guidelines and Contingency Factors for IT Governance". In: *European, Mediterranean & Middle Eastern Conference on Information Systems* (2012). DOI: 10.1145/2463728.2463789.
- [91] K. Petersen and C. Wohlin. "The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study". In: *Empirical Software Engineering* (2010). ISSN: 13823256. DOI: 10.1007/s10664-010-9136-6.
- [92] G. Plataniotis, S. De Kinderen, Q. Ma, and E. Proper. "A Conceptual Model for Compliance Checking Support of Enterprise Architecture Decisions". In: *Proceedings - 17th IEEE Conference on Business Informatics, CBI 2015*. 2015. ISBN: 9781467373401. DOI: 10.1109/CBI.2015.46.
- [93] E. Proper and D. Greefhorst. "The Roles of Principles in Enterprise Architecture". In: *Trends in Enterprise Architecture Research*. Ed. by E. Proper, M. M. Lankhorst, M. Schönherr, J. Barjis, and S. Overbeek. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 57–70. ISBN: 978-3-642-16819-2.
- [94] A. Qumer and B. Henderson-Sellers. "A framework to support the evaluation, adoption and improvement of agile methods in practice". In: *Journal of Systems and Software* (2008). ISSN: 01641212. DOI: 10.1016/j.jss.2007.12.806.
- [95] D. Radovanovic, T. Radojević, and M. Sarac. "IT audit in accordance with Cobit standard". In: *MIPRO, 2010 Proceedings of the 33rd International Convention*. 2010, pp. 1137–1141. ISBN: 9781424477630.
- [96] P. Resnick and R. E. Kraut. *Building Successful Online Communities: Evidence-Based Social Design*. 2009. ISBN: 9780262016575.
- [97] G. L. Richardson, B. M. Jackson, and G. W. Dickson. "A Principles-Based Enterprise Architecture: Lessons from Texaco and Star Enterprise". In: *MIS Quarterly* (1990). ISSN: 02767783. DOI: 10.2307/249787.
- [98] P. Rodríguez, J. Partanen, P. Kuvaja, and M. Oivo. "Combining lean thinking and agile methods for software development a case study of a finnish provider of wireless embedded systems". In: *Proceedings of the Annual Hawaii International Conference on System Sciences*. 2014. ISBN: 9781479925049. DOI: 10.1109/HICSS.2014.586.
- [99] R. Roos and J. C. Mentz. "Factors that influence enterprise architecture decision making". In: *2018 Conference on Information Communications Technology and Society (ICTAS)*. Mar. 2018, pp. 1–6. DOI: 10.1109/ICTAS.2018.8368763.
- [100] J. W. Ross, P. Weill, and D. C. Robertson. *Enterprise Architecture as Strategy*. 2006. ISBN: 1591398398.

- [101] D. Rost, B. Weitzel, M. Naab, T. Lenhart, and H. Schmitt. "Distilling Best Practices for Agile Development from Architecture Methodology". In: *Software Architecture*. Ed. by D. Weyns, R. Mirandola, and I. Crnkovic. Cham: Springer International Publishing, 2015, pp. 259–267. ISBN: 978-3-319-23727-5.
- [102] P. Runeson and M. Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* (2009). ISSN: 13823256. DOI: 10.1007/s10664-008-9102-8. arXiv: 9809069v1 [gr-qc].
- [103] V. Sambamurthy and R. Zmud. "Arrangements for Information Technology Governance: A Theory of Multiple Contingencies". In: *MIS Quarterly* (1999). ISSN: 02767783.
- [104] *Scaling Agile Framework*. URL: <https://www.scaledagileframework.com/>.
- [105] K. Schwaber and M. Beedle. *Agile Software Development with Scrum*. 2001. ISBN: 0130676349. DOI: 10.1109/2.947100.
- [106] Sein, Henfridsson, Puraio, Rossi, and Lindgren. "Action Design Research". In: *MIS Quarterly* (2011). ISSN: 02767783. DOI: 10.2307/23043488.
- [107] D. Šmite, N. B. Moe, A. Šablis, and C. Wohlin. "Software teams and their knowledge networks in large-scale software development". In: *Information and Software Technology* (2017). ISSN: 09505849. DOI: 10.1016/j.infsof.2017.01.003.
- [108] D. Stelzer. "Enterprise architecture principles: literature review and research directions". In: *Proceedings of the 2009 international conference on Service-oriented computing*. 2010. ISBN: 978-3-642-16131-5. DOI: 10.1007/978-3-642-16132-2.
- [109] T. Tamm, P. B. Seddon, G. Shanks, and P. Reynolds. "How does enterprise architecture add value to organisations?" In: *Communications of the Association for Information Systems* (2011). ISSN: 15293181. DOI: 10.1287/isre.3.1.60.
- [110] H. Tanriverdi. "Performance Effects of Information Technology Synergies in Multibusiness Firms". In: *MIS Quarterly* 30.1 (2006), pp. 57–77. ISSN: 02767783. DOI: 10.2307/25148717.
- [111] The Linux Foundation. *Cloud Native Computing Foundation*. 2018. URL: <https://www.cncf.io/>.
- [112] The Open Group. *TOGAF Version 9*. 2009. ISBN: 9789087532307. DOI: 10.1111/j.1365-2702.2009.02827.x.
- [113] J. F. Tripp, C. Riemenschneider, and J. B. Thatcher. "Job satisfaction in agile development teams: Agile development as work redesign". In: *Journal of the Association for Information Systems* (2016). ISSN: 1536-9323. DOI: 10.17705/1jais.00426.

- [114] J. Tripp, J. Saltz, and D. Turk. "Thoughts on Current and Future Research on Agile and Lean: Ensuring Relevance and Rigor". In: *Hawaii International Conference on System Sciences*. 2018. DOI: 10.24251/HICSS.2018.681.
- [115] D. Turk, R. France, and B. Rumpe. "Limitations of agile software processes". In: *Third International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP 2002)* (2002).
- [116] Ö. Uludağ, M. Hauder, M. Kleehaus, C. Schimpfle, and F. Matthes. "Supporting Large-Scale Agile Development with Domain-Driven Design". In: *Agile Processes in Software Engineering and Extreme Programming*. Ed. by J. Garbajosa, X. Wang, and A. Aguiar. Cham: Springer International Publishing, 2018, pp. 232–247. ISBN: 978-3-319-91602-6.
- [117] O. Uludağ, M. Kleehaus, X. Xu, and F. Matthes. "Investigating the Role of Architects in Scaling Agile Frameworks". In: *Proceedings - 2017 IEEE 21st International Enterprise Distributed Object Computing Conference, EDOC 2017*. 2017. ISBN: 9781509030453. DOI: 10.1109/EDOC.2017.25.
- [118] W. Van Grembergen and S. De Haes. *Implementing Information Technology Governance Models, Practices, and Cases*. New York: IGI Publishing, 2007. ISBN: 978-1599049243. DOI: 10.4018/978-1-59904-924-3.
- [119] VersionOne. *12th Annual State of Agile Survey Report*. 2018. URL: <https://stateofagile.versionone.com/%0A>.
- [120] J. Vlietland and H. Van Vliet. "Towards a governance framework for chains of Scrum teams". In: *Information and Software Technology*. 2015. DOI: 10.1016/j.infsof.2014.08.008.
- [121] X. Wang. "The Combination of Agile and Lean in Software Development: An Experience Report Analysis". In: *2011 AGILE Conference* (2011). DOI: 10.1109/AGILE.2011.36.
- [122] X. Wang, K. Conboy, and O. Cawley. "'Leagile' software development: An experience report analysis of the application of lean approaches in agile software development". In: *Journal of Systems and Software* (2012). ISSN: 01641212. DOI: 10.1016/j.jss.2012.01.061.
- [123] P. Webb, C. Pollard, and G. Ridley. "Attempting to define IT governance: Wisdom or folly?" In: *Proceedings of the Annual Hawaii International Conference on System Sciences*. 2006. ISBN: 0769525075. DOI: 10.1109/HICSS.2006.68.
- [124] P. Weill. "Don't just lead, govern: How top-performing firms govern IT". In: *MIS Quarterly Executive* (2004). ISSN: 14111128. DOI: 10.2139/ssrn.664612.

- [125] P. D. Weill and J. W. Ross. *ITSavvy: What top executives must know to go from pain to gain*. 2009.
- [126] P. Weill and J. W. Ross. "How Top Performers Manage IT Decisions Rights for Superior Results". In: *Harvard Business School Press* (2004). ISSN: 1556-5068. DOI: 10.2139/ssrn.664612.
- [127] R. Winter and R. Fischer. "Essential layers, artifacts, and dependencies of enterprise architecture". In: *Proceedings - 2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops, EDOCW2006*. 2006. ISBN: 076952558X. DOI: 10.1109/EDOCW.2006.33.
- [128] R. Winter and J. Schelp. "Enterprise architecture governance: the need for a business-to-IT approach". In: *Proceedings of the 2008 ACM symposium on ...* (2008). ISSN: 13506285. DOI: 10.1145/1363686.1363820.
- [129] P. Witt. "Corporate Governance im Wandel". In: *Zeitschrift Führung und Organisation* 69.3 (2000), pp. 159–163. ISSN: 0722-7485.
- [130] R. K. Yin. *Case Study Research: Design and Methods*. 2009. ISBN: 9781412960991.