Investigating the Role of Architects in Scaling Agile Frameworks

Ömer Uludağ, Martin Kleehaus, Xian Xu, Florian Matthes
Chair for Informatics 19
Technische Universität München (TUM)
D-85748, Garching
{*oemer.uludag,martin.kleehaus,xian.xu,matthes*}*@tum.de*

***Abstract*—This study describes the roles of architects in scaling agile frameworks with the help of a structured literature review. We aim to provide a primary analysis of 20 identified scaling agile frameworks. Subsequently, we thoroughly describe three popular scaling agile frameworks: Scaled Agile Framework, Large Scale Scrum, and Disciplined Agile 2.0. After specifying the main concepts of scaling agile frameworks, we characterize roles of enterprise, software, solution, and information architects, as identified in four scaling agile frameworks. Finally, we provide a discussion of generalizable findings on the role of architects in scaling agile frameworks.**

***Keywords*-scaling agile frameworks; agile software development; application architecture;**

I. Introduction

Enterprises struggle to deal with unpredictable competitive environments due to rapidly changing customer demands, regulatory changes, and technological advancements that can lead to the enterprise's success [1], [2]. Thus, the ability to detect relevant changes and respond in a timely and effective manner becomes an important determinant of the enterprise's survival [3]. Software development projects in such environments face changes either directly or indirectly. In order to face these challenges, the agile movement emerged in the 1990s, leading to the development of agile manifesto and many agile software development methods, including extreme programming (XP), kanban, and scrum [4]. With these agile methods, small, co-located, self-organizing teams work closely with the business customer on a single-project context, maximizing the customer value and quality of the delivered software product through rapid iterations and frequent feedback loops [4].

Since the initial application of these methods are tailored for small teams, large enterprises are interested in extending agile methods to include larger teams and inter-team coordination and communication [5]. Various scaling agile frameworks, e.g., the Scaled Agile Framework (SAFe), Disciplined Agile 2.0 (DA 2.0), and Large-Scale Scrum (LeSS), were proposed to resolve issues associated with team size, customer involvement, and project constraints [6].

Agile methods imply that the architecture should evolve incrementally and constantly be tested with known requirements rather than being imposed by some direct structuring force (emergent architecture design) [7]. The practice of emergent architecture design is effective at the team level but insufficient when developing large systems. It causes excessive redesign efforts, architectural divergence, and functional redundancy increasing the complexity of application architectures [7], [8]. Therefore, an intentional architecture design is required, which embraces architectural initiatives and guidance for inter-team design and implementation synchronization [7], [9]. The effective evolution of an application's architecture requires the right balance of emergent and intentional architecture design. This balance is an essential determinant for addressing the complexity of large applications architectures and large-scale agile projects [7], [9].

However, literature documenting the influences of scaling agile frameworks on application architectures and the involvement of architects in large-scale agile projects is still scarce. The main objective of this study is to investigate the role of architects in scaling agile frameworks. Based on this objective, three research questions (RQ) were formulated

- *RQ1: What are the types of scaling agile frameworks?*
- *RQ2: What are the roles of architects in scaling agile frameworks?*
- *RQ3: What are the generalizable findings about the role of architects in scaling agile frameworks?*

A. Research methodology

To identify material relevant to stated goal of this study, we applied a structured literature review approach as recommended by Brocke et al. [10]. In the first phase, we defined the scope of the review and identified suitable research questions about the role of architects in scaling agile frameworks. In the second phase, key concepts were identified by concept mapping, which also provided the opportunity to identify additional relevant search terms: *Scaled Agile Organization*, *Enterprise Architecture*, *Scaled Agile Enterprise Architecture*, *Scaled Agile Framework*, *Software Engineering*, and *Scaled Agile Software Engineering*, together with a variety of related concepts, synonyms, and homonyms. These were applied to the subsequent literature search in the third phase. We examined a range of different Information Systems journals, conference proceedings, documentations, and books using EBSCOhost, ScienceDirect, Scopus, ACM Digital Library, IEEExplore, SpringerLink, Emerald Insight, and Google Scholar. Having compiled the aforementioned list of search terms, we then used them in electronic full-text search queries. In total, we obtained 146 relevant sources. In the fourth phase, we created a concept matrix juxtaposing the different architect roles with the identified scaling agile frameworks to investigate their roles within the scaling agile frameworks. In the last phase, the comparative analysis of architect roles resulted in generalizable findings on the role of architects in scaling agile frameworks.

The remainder of this paper is structured as follows. In Section II, we define large-scale agile developments, providing a primary analysis of the scaling agile frameworks we identified in the literature and describing thoroughly the three most mature frameworks. In Section III, we analyze architect roles identified in scaling agile frameworks. We analyze and discuss these findings in Section IV before concluding the paper with remarks on future research.

II. Scaling Agile Frameworks

The origin of agile ideas in business started with the creation of the Agile Consortium in 1994 [11]. These ideas were discovered independently in software engineering, culminating in the creation of the Agile Manifesto [12] but have exactly the same guiding principles as those in business.

The development of scaling agile frameworks dates back to 1992, with the Crystal Family [13], and has increasingly gained in popularity in the last few years. The main purpose of such frameworks is to manage large agile teams with more than 50 developers distributed across multiple geographical locations in an agile way. Traditional agile methods such as Scrum are not capable of managing this number of developers. However, scaling agile methods introduce new challenges, such as inter-team coordination and distribution of work without a defined architecture or properly defined requirements [14]. We further define the term large-scale agile development and provide a discussion of the frameworks that currently attract the most attention.

A. Definition of large-scale agile development

The difficulty of introducing agile methods increases with organization size [15]. The adoption of agile frameworks often requires changing the entire organizational culture. Larger organizations have more dependencies between projects and teams, which slows down any organizational change. Large size also increases the need for formal documentation, which in turn reduces agility [16]. In addition to inter-team coordination, agile teams also have to interact with other organizational units, which are typically non-agile in nature. Therefore, several companies have devised new approaches to scaling agile methods to projects wherein a lot of people are involved. Hence, the term *large-scale agile development*[1]. refers to agile development in everything from large teams to large multi-team projects that want to make use of agile development principles at the portfolio or enterprise level [18]. Dingsøyr et al. [18] identified that large scale agile projects had been regarded in terms of the number of people or teams, project budget, code base size, or project duration. Examples of cases that were described as *large-scale* covered projects costing over $10 million with teams of more than 50 people, code bases consisting of over half a million lines of code [19], durations of 2 years, and scopes of 60–80 features [20]. Based on their findings, Dingsøyr et al. [18] measured large-scale projects by the number of collaborating and coordinating teams as large-scale projects with 2–9 collaborating teams and very large-scale projects with over 10 collaborating teams.

[1]A primary systematic literature review on the challenges and success factors for large-scale agile development was conducted by [17].

B. Introduction to the most popular frameworks

The structured literature review described in Section I-A revealed 20 scaling agile frameworks listed in Table I. Most of the frameworks emerged from very basic approaches to agile development, namely XP and Scrum, and were enhanced as necessary in order to be applicable to very large multi-team projects. In Table I, we summarize our findings and provide primary information about the methodologists who invented and published the frameworks, the organizations which were built upon them, and the scaling agile approaches involved. The maturity section of Table I indicates how well-established the particular framework is. Here, we calculate the maturity[2] of a framework based on

- the number of paper contributions that we found in the literature search;
- the number of case studies described on the homepage of the regarded framework;
- the available documentation that could be found either on its homepage or in other sources;
- the training courses and certifications offered by the organization; and
- the content of the community forums and blogs wherein the companies shared their knowledge and other information about the framework.

The LeSS, SAFe, and DA 2.0 frameworks are especially mature, as they are cited very often in the literature, describing many real-world use cases, and also fulfill the other adoption criteria. Therefore, we will subsequently provide more details about these popular frameworks:

LeSS was released in 2008 by Craig Larman and Bas Vodde and extends Scrum with scaling rules and guidelines without losing sight of Scrums's original goals. LeSS specifies additional organizational changes, which are not addressed in traditional Scrum. For instance cross-functional,

[2]The maturity is calculated based on the sum of the equally weighted maturity criteria. Particularly, 'Yes' values are coded as '1', whereas 'No' values are coded as '0'. Contributions and cases values of each framework are divided by the max. values of contributions and cases. For instance, the maturity rating of Crystal Family is calculated as follows: $\frac{17}{35} \cdot 0.2 + \frac{1}{35} \cdot 0.2 + 1 \cdot 0.2 + 0 \cdot 0.2 + 1 \cdot 0.2 = 0.503 \rightarrow$ ◑.

Table I
PRIMARY ANALYSIS OF SCALING AGILE FRAMEWORKS.

	Descriptive Information				Maturity					
	Methodologist	Organization	Publication Date	Category	Contri-butions	Cases	Docu-mentation	Training Courses and Certifications	Community, Forum or Blog	Rating
Crystal Family	Alistair Cockburn	-	1992	Set of Methods	17	1	Yes	No	Yes	◐
Dynamic Systems Development Method Agile Project Framework for Scrum	Arie van Bennekum	DSDM Consortium	1994	Framework	28	4	Yes	Yes	Yes	◕
Scrum-of-Scrums	Jeff Sutherland and Ken Schwaber	Scrum Inc.	2001	Mechanism	27	2	Yes	No	Yes	◐
Enterprise Scrum	Mike Beedle	Enterprise Scrum Inc.	2002	Framework	4	-	Yes	Yes	Yes	◐
Agile Software Solution Framework	Asif Qumer and Brian Henderson-Sellers	University of Technology	2007	Framework	2	2	No	No	No	○
Large Scale Scrum	Craig Larman and Bas Vodde	LeSS Company B.V.	2008	Framework	29	22	Yes	Yes	Yes	●
Scaled Agile Framework	Dean Leffingwell	Scaled Agile Inc.	2011	Framework	35	35	Yes	Yes	Yes	●
Disciplined Agile 2.0	Scrott Ambler	Disciplined Agile Consortium	2012	Framework	27	4	Yes	Yes	Yes	●
Spotify Model	Henrik Kniberg, Anders Ivarsson, and Joakim Sundén	Spotify	2012	Model	11	1	Yes	No	Yes	◔
Mega Framework	Rafael Maranzato, Marden Neubert, and Paula Heculano	Universo Online S.A	2012	Framework	2	1	No	No	No	○
Enterprise Agile Delivery and Agile Governance Practice	Erik Marks	AgilePath	2012	Set of Practices	1	-	Yes	No	Yes	◔
Recipes for Agile Governance in the Enterprise	Kevin Thompson	Cprime	2013	Framework	4	1	Yes	Yes	No	◔
Continuous Agile Framework	Andy Singleton	Maxos LLC	2014	Framework	3	-	Yes	No	Yes	◔
Scrum at Scale	Jeff Sutherland and Alex Brown	Scrum Inc.	2014	Framework	9	-	Yes	Yes	Yes	◐
Enterprise Transition Framework	-	agile42	2014	Framework	1	2	Yes	Yes	Yes	◐
ScALeD Agile Lean Development	Peter Beck, Markus Gärtner, Christoph Mathis, Stefan Roock and Andreas Schliep	-	2014	Set of Principles	2	-	Yes	No	Yes	◔
eXponential Simple Continuous Autonomous Learning Ecosystem	Peter Merel	Xscale Alliance	2014	Set of Principles	3	-	Yes	Yes	Yes	◐
Lean Enterprise Agile Framework	-	LeanPitch Technologies	2015	Framework	0	-	Yes	Yes	Yes	◐
Nexus	Ken Schwaber	Scrum.org	2015	Framework	5	-	Yes	Yes	Yes	◐
FAST Agile	Ron Quartel	Cron Technologies	2015	Set of Methods	2	-	Yes	No	Yes	◔

cross-component, end-to-end feature teams are introduced by LeSS through the elimination of traditional team lead and project manager roles [21].

Larman and Vodde proposed two different frameworks depending on the size of the project. The basic LeSS framework provides guidelines and techniques for agile development with less than ten teams [6].

In basic LeSS, a single product owner (PO) is common to all ten teams, but no other special roles are specified compared with standard Scrum. The LeSS framework changes the structure of sprint planning meetings compared with the traditional Scrum approach. Here, each of the agile teams, represented by two members per team, plus the one overall PO decide which chunk of product backlog items to work on. This is in contrast with standard Scrum wherein the rest of the agile team also participates. When contention occurs over a backlog item, the PO mediates between teams. Likewise, sprint review changes to a single meeting for all agile teams. However, it is limited to two team members per agile team. In addition, three more changes are established [6]

- The *inter-team coordination* meeting is designed to increase information sharing and coordination. It can be conducted frequently during the week and can take various forms, including open space, town hall meeting, multi-team daily Scrum, or Scrum of Scrums formats.
- The *joint light product backlog refinement* meeting focuses on refining product backlog items for upcoming sprints. It is restricted to two representatives per team and should not exceed 5% of the sprint duration.
- Finally, a *joint retrospective* is added that aims to identify and plan improvement experiments for the overall product or organization. The PO, scrum master, and one representative from each team attend this meeting.

Agile development with more than ten teams is guided by the second LeSS framework, named LeSS Huge. It introduces additional scaling elements, which are required to manage hundreds of developers in large enterprises. LeSS Huge introduces a new concept, namely, requirement areas (RAs). RAs encompass major areas of customer concern from a product point of view and may grow or shrink over time in order to match product needs. All RAs follow the same sprint cadence and aim for continuous integration across the entire product. Adding the RA as an attribute in the product backlog creates an area product backlog (APB) view for each RA. This represents a new feature in LeSS Huge. Each product backlog item belongs to one area backlog. Area backlog items are defined, prioritized, and split, as needed, by the area product owner (APO). The APO focuses on one APB and is usually a specialist in that area. The APO acts similar as the PO would in the smaller LeSS framework.

SAFe was released in 2011 by Dean Leffingwell and is now at version 4.0. SAFe builds on existing lean and agile principles that are combined into a method for large-scale agile projects. SAFe provides a soft introduction to the agile world as it specifies many structured patterns. This is often needed for those who are transitioning from a more traditional environment, particularly in the context of a large project. A common problem with agile adoption is the difficulty in introducing such a major cultural change to an organization. Thus, SAFe provides the structure needed to make the transition more predictable, even though it follows agile practices and stresses autonomy and decision-making for knowledge workers. SAFe highlights four levels of organization: team, program, value stream, and portfolio. Each level integrates agile and lean practices, manages its own activities, and is aligned with the other levels.

At the team level, the techniques outlined are those used in Scrum, and two-week sprint cycles are recommended. Each team comprises 5–9 members and has a scrum master and a PO, similar to standard Scrum. All SAFe teams are part of one agile release train (ART), a team of agile teams that delivers a continuous flow of incremental releases of value. Each agile team is responsible for defining, building, and testing stories from its team backlog in a series of iterations using common iteration cadences and synchronization to align its activities with other teams so that the entire system is iterating in unison. Teams use ScrumXP or Kanban to deliver prototypes every two weeks [22].

At the program level, SAFe extends Scrum using the same ideas but on a higher level. The program level is based on an ART, which is composed of five sprint cycles. There is also a sixth innovation planning sprint, which allows teams to innovate, inspect, and adapt. Teams, roles, and activities are organized around the ART [22]. At this level, a product manager (PM) serves as the content authority for the ART and is accountable for identifying program backlog priorities. In addition, the PM works with POs to optimize feature delivery and direct the work of POs at the team level. A release train engineer (RTE) facilitates program level processes and execution, escalates impediments, manages risk, and helps to drive continuous improvement.

At the optional value stream level, the value stream engineer (VSE) plays a similar role, facilitating and guiding the work of all ARTs and suppliers. Further important roles, such as business owner, DevOps team member, release manager, and solution manager have been described in [22].

SAFe also specifies processes at one higher level, the portfolio level, using lean principles, such as optimizing value streams, which are long-lived series of steps used to deliver value. These help executives and leaders identify and prioritize epics and features that can be broken down at the program level and scheduled for ARTs [22].

The **DA 2.0** framework, previously kwown as Disciplined Agile Delivery, was released in 2012 by Scott Ambler and Mark Lines [23]. In comparison with SAFe, DA 2.0 aims to address areas that are not thoroughly covered in smaller scaling agile frameworks and recommends three phases: inception, construction, and transition. While many agile frameworks address what DA 2.0 calls the construction phase, DA 2.0 provides recommendations for processes that come both earlier in the project (inception) and as teams prepare for delivery (transition). DA 2.0 also provides flexibility by suggesting different process guidelines for four categories of life cycles: agile/basic, lean/advanced, continuous delivery, and exploratory.

The construction phase of agile/basic is Scrum, whereas the lean/advanced life cycle uses processes similar to Kanban. The inception phase is used to stock a work item pool that is organized to achieve business values, fixed delivery dates, expedited delivery, or some other intangible goal. During the transition phase, planning, retrospection, prototyping, stand up meetings, and other activities are undertaken. The continuous delivery life cycle focuses on mature DevOps, continuous integration, and deployment processes for projects that require frequent delivery to stakeholders. The exploratory life cycle minimizes early planning in favor of fast delivery, gaining feedback, and incorporating that feedback into the next delivery [23].

In the third continuous delivery life cycle, the inception phase is explicit and has a very brief transition period. In this life cycle, products are produced on a very regular basis: daily, weekly, or monthly [23].

The last life cycle, the exploratory life cycle, aims to encourage agile teams to put themselves in start-up or research situations wherein the stakeholders have clear ideas for a new product but do not yet understand the needs of their user base [23].

Table II
IDENTIFIED ARCHITECT ROLES IN SCALING AGILE FRAMEWORKS.

	Enterprise Architect	Software Architect	Solution Architect	Information Architect
DSDM	-	X	X	-
SAFe	X	X	X	X
DA 2.0	X	X	X	-
EADAGP	X	-	-	X

III. The Role of Architects in Scaling Agile Frameworks

Since traditional agile methods, such as XP or Scrum, do not include the role of architects, we recognize that this is no longer valid for scaling agile frameworks. In particular, we have seen that several scaling agile frameworks involve various architect roles. We have selected a set of predominant architect roles relevant to the realm of enterprise architecture (EA) from refs. [24] and [25] to describe their function in scaling agile frameworks. These roles comprise the *enterprise architect*, *software architect*, *solution architect*, and *information architect*[3]. We have searched each relevant source for these architect roles and related them to the different scaling agile frameworks. The results are summarized in Table II.

Only 4 out of 20 scaling agile frameworks include architect roles, namely Dynamic Systems Development Method Agile Project Framework for Scrum (DSDM), SAFe, DA 2.0, and Enterprise Agile Delivery and Agile Governance Practice (EADAGP). The findings also reveal that no framework describes all architect roles in detail. The frameworks predominantly consider the roles of enterprise, software, and solution architects. In addition, the results show that more mature frameworks are more likely than others to describe architect roles. However, due to limited information and unavailable documentation, further analyses are required to emphasize this assumption.

We have created the following role description template, based on refs. [24], [26], [27] to describe architect roles

- *Key concerns* describe the key interests of the architect role.
- *Area of interests* describes the area of interests of the architect role, e.g., special project tasks.
- *Contributions* describe the contributions of the architect role, e.g., resources like work, financial capital, or other types of engagements.
- *Strategies* describes the appropriate strategies for working with the architect role from management view.
- *Responsibilities* describes the responsibilities of the architect role.
- *Commitments* describes the commitments of the architect role, which can be either active opposition, passive opposition, neutral, passive support, or active support.

The following sections present our results about the role of architects in various scaling agile frameworks.

A. The role of enterprise architect

The role of enterprise architect is only considered by SAFe, DA 2.0, and EADAGP.

In **SAFe**, an enterprise architect works with business stakeholders and software and solution architects to drive holistic technology implementation across value streams.
The enterprise architect is *concerned* with driving EA strategy, which comprises five key aspects, namely choice of technology, software and solution strategy, development and deployment infrastructure strategy, inter-program collaboration, and implementation strategy. This is communicated, along with other key business drivers of architecture, to system architects and nontechnical stakeholders.
The main *contributions* of the enterprise architect are providing strategic technical directions and driving collaboration of programs and teams around a common technical vision.
The portfolio level represents the enterprise architect's *area of interest*.
The *strategy* for working with enterprise architects is to involve them actively in the portfolio level by ensuring the presence of enterprise-wide architectural systems, platforms, and infrastructures.
The key *responsibilities* of the enterprise architect are

- maintaining a high-level and holistic vision of enterprise solutions and development initiatives;
- understanding and communicating strategic themes and other key business drivers for architecture to system architects and nontechnical stakeholders;
- working with business stakeholders and software and solution architects to drive holistic technology implementation across value streams;
- working closely with software and solution architects to ensure that individual program and product strategies align with enterprise objectives;
- participating in the strategy for building and maintaining the enterprise architectural runway; and
- facilitating the reuse of ideas, components, and patterns.

The *commitment* of the enterprise architect is the active support of agile teams [22].

Enterprise awareness is one of the key aspects of the **DA 2.0** framework. Enterprise awareness motivates agile teams to consider the overall needs of the organization and to leverage existing assets in alignment with an enterprise-level strategy. DA 2.0 recommends that agile teams work closely with enterprise professionals, such as enterprise architects.
Within DA 2.0, the enterprise architect has both a primary role as a stakeholder and a secondary role as a specialist for assisting agile teams. The *key concerns* of enterprise architects are addressing strategies for supporting delivery teams and other stakeholders, evolving and capturing the EA, and governing the EA efforts.
The inception, construction, and ongoing process goals of the DA 2.0 framework form the *area of interest* of the enterprise architect. In the inception phase, the enterprise architect works closely with agile teams to align them with enterprise goals and to provide non-functional requirements (NFRs). Further, the enterprise architect supports agile teams during the initial architectural envisioning and modeling efforts and

[3]We have also considered synonyms for each architect role, e.g., domain architect denotes the same role as solution architect.

the initial technical strategy definition phase by ensuring that they leverage as much of the existing infrastructure as possible. In the construction phase, the enterprise architect collaborates with agile teams to ensure that their solution reflects the overall strategy of the organization. Within the ongoing process goal, the enterprise architect holds regular coordination meetings with product management teams to ensure consistency and manage dependencies across teams.
The enterprise architect *contributes* to software development by providing guidance to agile teams by producing high-level technology and business roadmaps, which capture the organization's vision and by helping agile teams to understand the overall vision. The *strategy* for working with enterprise architects is to involve them passively in development projects.
The key *responsibilities* of the enterprise architect are

- supporting and collaborating closely with stakeholders on a regular basis to understand their needs and to develop the organization's roadmap;
- supporting and collaborating closely with agile teams on a regular basis to guide them through the business and technical roadmaps and help them to identify potentially reusable assets and technical debts;
- negotiating technical dependencies between solutions;
- exploring architectural views; and
- adopting and tailoring architectural frameworks.

The *commitment* of the enterprise architect is the passive support of agile teams by providing guidance and roadmaps [23], [28], [29], [30], [31], [32].

The **EADAGP** framework includes an event-driven governance model, which provides a lightweight, lean, and virtual governance model design. EADAGP introduces a governance buffer zone, which aims to protect agile teams from the slowness, friction, and rigidity of traditional IT governance models. The agile governance buffer zone is a subtractive layer that is realized by combining three different styles of governance, namely top-down prescriptive governance (traditional IT governance), community governance, and self-governance. The enterprise architect forms, along with scrum master(s), PO(s), and the agile governance owner the agile community team (ACT). This is a community governance construct, responsible for governing multiple agile teams that are aligned with one or more releases.
Within the ACT, the enterprise architect is *concerned* with governance requirements that span multiple sprints or releases and cross-sprint team coordination and collaboration. Community governance, in particular the ACT, represents the *area of interest* of the enterprise architect.
The main *contributions* of the enterprise architect are addressing governance requests, escalating them to IT governance if they cannot be addressed, and notifying agile teams and IT governance for their agreement and/or approval.
The *strategy* for working with enterprise architects is to involve them passively within the ACT so they can make appropriate governance decisions and provide guidance.
The key *responsibilities* of the enterprise architect are

- escalating governance issues to IT governance;
- supporting agile teams by making appropriate governance decisions and providing guidance;
- communicating governance decisions to enterprise IT governance for their agreement;
- supporting the governance backlog grooming process;
- harmonizing governance requirements across sprints and agile teams;
- reporting technology and architecture requirements/issues to EA oversight for alignment and issue resolution;
- identifying security requirements and challenges that may not have been pre-determined; and
- raising potential compliance and risk requirements that have to be reviewed and signed off by governance, risk and compliance bodies, and EA sign offs.

The *commitment* of the enterprise architect is the passive support of agile teams by protecting them from the slowness and rigidity of traditional IT governance [33], [34].

B. The role of software architect

DSDM, SAFe, and DA 2.0 include the role of software architects.

DSDM does not explicitly prescribe the role of software architect but it does specify the role of a technical coordinator, whose responsibilities can be allocated to more than one person, e.g., a system architect. Thus, we will concentrate on describing the role of the technical coordinator, which is relevant to the role of a software architect.
As the project's technical authority, the software architect's *key concern* is to ensure that the project is technically coherent and meets the desired technical standards. The software architect holds the business and technical visions for the project.
The project level constitutes the *area of interest* of the software architect. Therein, the software architect may be a part of a project board or steering committee for the project.
Technical development and direction of the solution and system architecture definitions are the software architect's main *contributions*.
The *strategy* for working with software architects is to involve them as technical coordinators.
The *responsibilities* of the software architect are

- agreeing and controlling the technical architecture;
- identifying and owning architectural and other technically based risks;
- working with business analysts to evaluate the technical options and decide on the best way to turn the high-level business requirements into a technical solution;
- promoting standards for technical best practice;

- acting as the final arbiter of technical differences between agile team members; and
- defining the system architecture that constitutes the technical framework within which the solution will be developed and providing a high-level description of the structure of that solution.

The *commitment* of the software architect is passive support of agile teams [11], [35], [36].

SAFe's key roles at the program level include the RTE, product management, and system architect/engineer, hereafter the software architect. The software architect role is filled by an individual or small team that has technical responsibility for the overall architectural and engineering design of the system and aligns ARTs with the common technical and architectural vision for the solution under development. The software architect participates in defining the system, as well as any subsystems and interfaces, validating technology assumptions, and evaluating alternatives. The software architect works at a higher level of abstraction than the teams and supports system development by providing, communicating, and evolving the larger technological and architectural view of the solution.
The system architect is *concerned* with executing the upfront architecture design, guiding the emergent architecture for all program teams, and defining the architectural runway that supports new feature development, as well as providing guidance for common solution behaviors, shared components, and separation of concerns.
The program level, particularly the ART, represents the software architect's main *area of interest*.
The software architect *contributes* to software development by working with agile teams and providing technical enablement with respect to subsystems and capability areas under the purview of the ART.
The *strategy* for working with software architects is to involve them at program level.
The key *responsibilities* of the software architect are

- defining NFRs, major system elements, subsystems, and interfaces;
- preparing the architecture vision briefing within the program increment (PI) planning event;
- presenting the architecture vision, which may include descriptions of new architectural epics for common infrastructure, any large-scale refactors under consideration, and system-level NFRs; and
- supporting the PO by refining the team backlog.

In addition, the software architect shares several common responsibilities with the solution architect, which are

- defining and refining NFRs, subsystems, and interfaces to ensure that the solution meets relevant standards and other system quality requirements;
- defining subsystems and their interfaces, allocating responsibilities to subsystems, understanding solution deployment, and communicating requirements for interactions with the solution context;
- preparing for the PI planning event by updating enabler definitions and models;
- assisting with decision-making and sequencing of the key technological infrastructures that will host the new business functionality;
- creating and supporting enabler epics by steering them through the Kanban system, providing both the guidance needed to analyze them and the information needed to estimate and implement them;
- working with portfolio stakeholders, particularly the enterprise architect, to develop, analyze, split, and realize the implementation of enabler epics; and
- planning and developing the architectural runway in support of upcoming business features and capabilities.

The *commitment* of the software architect is the active support of agile teams within the program level [22].

As already mentioned in Section II-B, **DA 2.0** introduces the AM role of architecture owner, which is typically the software or solution architect, hereafter the software architect[4]. The software architect is *concerned* with owning architecture decisions for the team and facilitating the creation and evolution of the overall solution design.
The first two phases, namely the inception and transition phases, as well as the ongoing process goals that occur throughout the delivery life cycle represent the *area of interest* of the software architect. The software architect is involved during the inception phase by contributing to exploration of the initial solution requirements, alignment of the solution with both the business and technical directions of the organization, and identification of the initial technical strategy. During the construction phase of the solution, the software architect updates the architectural handbook in the iteration in which the features are delivered. The software architect contributes to the goal of activity coordination within the ongoing process goal by having regularly scheduled information meetings with the PO to share information about work details, priorities, dependencies, and issues. The software architect mainly *contributes* to the identification of the initial technical strategy, definition of the architecture, and development of technical aspects of the overall solution architecture. The *strategy* for working with software architects is to involve them during the two first life cycle phases and with ongoing process goals of DA 2.0.
The key *responsibilities* of the software architect are

- guiding the creation and evolution of the solution architecture that the team is working on;
- mentoring and coaching other team members with best practices of architecture;
- understanding the architectural direction and standards

[4]Since DA 2.0 does not differentiates between the roles of software architect and solution architect, we have merged the findings for both roles.

of the organization and helping to ensure that the agile team adheres to them appropriately;
- understanding existing enterprise assets such as frameworks, patterns, and subsystems and ensuring that the team uses them where appropriate;
- ensuring that the solution will be easy to support by encouraging good design and refactoring to minimize technical debt;
- ensuring that the solution is integrated and tested on a regular basis, ideally via continuous integration;
- working closely with the team lead to identify technical risks in the project and determining strategies to mitigate them; and
- leading the initial architecture envisioning effort at the beginning of the project and supporting the initial requirements envisioning effort.

The *commitment* of the software architect is the active support of agile teams [23], [31], [32].

C. The role of solution architect

The role of solution architects is supported by DSDM, SAFe, and DA 2.0.

As discussed previously in Section III-B, **DSDM** does not include the architect roles explicitly. However, a solution architect can also be allocated to the role of a technical coordinator. Thus, we will describe the technical coordinator characteristics that are relevant to the solution architect role.
The solution architect is *concerned* with definition of the solution architecture.
Thus, the solution architecture represents the solution architect's *area of interest*. Within DSDM, the solution architecture is set during the foundation phase, which makes a preliminary investigation of the feasibility of the solution and establishes a fundamental understanding of the business rationale for the project and the potential solution. The solution architecture constitutes an evolutionary product, which provides a high-level design framework for the solution. It covers both business and technical aspects of the solution.
The main *contribution* of the solution architect is in the definition of the solution architecture during the foundation phase of the project.
The *strategy* for working with solution architects is to involve them actively in defining the solution architecture during the feasibility, foundation, evolutionary development, and deployment phases.
The solution architect is *responsible* for the overall design and integrity of the technical aspects of the solution, which comprise the solution architecture.
The *commitment* of the solution architect is the active support of agile teams by providing the definition of the solution architecture [11], [35], [36].

At the value stream level, **SAFe** introduces additional roles, namely solution management, VSE, and solution architect/engineer, hereafter solution architect. The role of solution architect is filled by cross-disciplinary teams that take a systemwide view of solution development.
The solution architect is *concerned* with the overall architectural design of the solution, definition of the higher-level functional and NFRs, determination of major components and subsystems, and definition of interfaces.
The primary *area of interest* of the solution architect is the value stream level.
As a technical leader, the solution architect *contributes* to the entire solution under development by communicating and evolving the larger technological and architectural view of the solution and aligning the value stream and ARTs to a common technological and architectural vision.
The *strategy* for working with solution architects is to involve them actively and to enable their close collaboration with business stakeholders, teams, customers, suppliers, and third-party stakeholders.
Since the solution architect shares several common responsibilities with the software architect in SAFe, we will only describe responsibilities, which are exclusive to the solution architect (see Section III-B for common responsibilities).
The exclusive *responsibilities* of the solution architect are
- supporting the solution management by managing the value stream kanban;
- discussing upcoming enabler capabilities and epics with the solution management;
- defining the overarching architecture that connects the solution across ARTs;
- working with the system architect to guide the architecture developed by the ARTs;
- ensuring technical alignment with the solution context, including interfaces and constraints;
- attending to the value stream and ART PI planning events; and
- updating progress toward milestones, program PI objectives, and dependencies among the ARTs

The *commitment* of the solution architect is the active support of agile teams at value stream level [22].

D. The role of information architect

The role of information architect is only supported by SAFe and EADAGP.

Within **SAFe**, the *key concern* of the information architect is to participate in a shared services role in order to support development by quickly bringing specialized expertise to bear on areas of the system or solution that require unique knowledge and skills.
The *area of interest* of the information architect is the ART and value stream.
The main *contribution* of the information architect is to support agile teams on the ART and to contribute to the architectural runway. Along with a system architect, a scrum master and one or two agile teams. They create the technological infrastructure to support the highest-priority features

in a near-term PI and the intentional architecture that guides cross-team design and implementation synchronization.

The *strategy* for working with information architects is to involve them actively in the project and to embed them periodically in agile teams.

The *responsibilities* of the information architect are

- participating in PI planning as well as in pre- and post-PI planning;
- driving requirements and taking ownership of dependent backlog items;
- collaborating with agile teams to fulfill dependencies during PI executions; and
- participating in system and solution demos.

The *commitment* of the information architect is the active support of agile teams [22].

Within the **EADAGP** framework, the *key concerns* of the information architect is participation within the ACT. The information architect works on governance requirements that cut cross multiple sprint teams or releases.

The *area of interest* of the information architect is the ACT.

The main *contributions* of the information architect is providing information architecture governance requirements that span multiple sprints or releases.

The *strategy* for working with information architects is to involve them passively in the project.

The key *responsibilities* of the information architect are

- escalating and communicating governance issues and decisions to IT governance;
- supporting agile teams by making appropriate governance decisions and providing guidance;
- supporting the governance backlog grooming process;
- harmonizing governance requirements across sprints and agile teams;
- surfacing technology and architecture requirements;
- identifying security requirements and challenges that may not have been pre-determined; and
- raising potential compliance and risk issues.

The *commitment* of the information architect is the passive support of agile teams and, thus, of software development since the information architect is only responsible for imposing governance requirements on agile teams [33].

IV. Discussion

We now discuss the main outcomes of our findings. ***Increasing development speed by balancing emergent and intentional architecture design.*** While some scaling agile frameworks, e.g., LeSS, are against upfront architecture design, other frameworks, such as DA 2.0 and SAFe, endorse upfront architecture design and planning. While they highlight the dangers of traditional architectural habits, e.g., heavyweight documentations, tedious upfront design approaches, and imposed architectural guidelines, they also realize that some initial envisioning performed at the beginning of the project can increase effectiveness and reduce excessive redesign efforts as the teams are steered in the intended direction. Only a close collaboration between agile teams, software, solution, and enterprise architects can enable an optimal interplay of emergent design and intentional architecture.

Finding the right balance between centralized and decentralized architectural decision-making. Escalating any type of architectural decision to higher levels of authority increases delay and decreases the usefulness of the decision. A balanced combination of centralized and decentralized decision-making provides many benefits, e.g., faster time to market and higher-quality products and services.

Sparing agile development from traditional IT governance. Some scaling agile frameworks, such as DSDM and DA 2.0, recognize the real value of agility in terms of project productivity and solution quality while acknowledging and accepting necessary constraints, e.g., governance, architecture, and infrastructure strategies, which often exist when working in a corporate environment. However, traditional governance models are slowing down agile teams in large organizations. SAFe, EADAGP, and DA 2.0 recommend new governance models that are collaborative, decentralized, and light-weight. These new models allow teams to decentralize their decision making and to govern themselves.

Ensuring the reuse of enterprise assets. Architects are aware of existing enterprise assets, e.g., patterns and standards, which are available for reuse, and ensure that agile teams utilize them where applicable. This accelerates the development process and reduces time to market.

V. Conclusion and future work

In this study, we have motivated the need for scaling existing agile methods to large-scale agile development due to their deficiencies in inter-team coordination and communication. We have then presented a primary analysis of the identified scaling agile frameworks. Based on our maturity assessment, we have identified LeSS, SAFe, and DA 2.0 as the most mature frameworks. Finally, we have extensively characterized the different architect roles that have been identified in scaling agile frameworks. Our findings indicate that architects both actively and passively support agile teams by driving architectural initiatives, participating in architectural runways, harmonizing governance requirements, and ensuring technical alignment in solution contexts. Future research may analyze the challenges faced by architects in scaling agile environments by conducting case studies in organizations that can provide practical experience of adopting scaling agile frameworks.

References

[1] P. Weill and S. Woerner, “Thriving in an increasingly digital ecosystem,” *MIT Sloan Management Review*, vol. 56, no. 4, p. 27, 2015.

[2] B. Sherehiy, W. Karwowski, and J. Layer, "A review of enterprise agility: Concepts, frameworks, and attributes," *International Journal of industrial ergonomics*, vol. 37, no. 5, pp. 445–460, 2007.

[3] E. Overby, A. Bharadwaj, and V. Sambamurthy, "Enterprise agility and the enabling role of information technology," *European Journal of Information Systems*, vol. 15, no. 2, pp. 120–131, 2006.

[4] P. Kettunen, "Extending software project agility with new product development enterprise agility," *Software Process: Improvement and Practice*, vol. 12, no. 6, pp. 541–548, 2007.

[5] M. Alqudah and R. Razali, "A review of scaling agile methods in large software development," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, no. 6, pp. 28–35, 2016.

[6] A. Vaidya, "Does dad know best, is it better to do less or just be safe? adapting scaling agile practices into the enterprise," *PNSQC. ORG*, pp. 828–837, 2014.

[7] "Agile architecture," http://www.scaledagileframework.com/agile-architecture/, accessed: 2017-04-26.

[8] M. Mocker, "What is complex about 273 applications? untangling application architecture complexity in a case of european investment banking," in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*. IEEE, 2009, pp. 1–14.

[9] M. Waterman, "Reconciling agility and architecture: A theory of agile architecture," Ph.D. dissertation, Victoria University of Wellington, 2014.

[10] J. Vom Brocke, A. Simons, B. Niehaves, K. Riemer, R. Plattfaut, and A. Cleven, "Reconstructing the giant: On the importance of rigour in documenting the literature search process," in *ECIS*, vol. 9, 2009, pp. 2206–2217.

[11] A. B. Consortium, *The DSDM Agile Project Framework Handbook*. Agile Business Consortium, 2014.

[12] M. Fowler and J. Highsmith, "The agile manifesto," *Software Development*, vol. 9, no. 8, pp. 28–35, 2001.

[13] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: review and analysis," VTT Technical report, Tech. Rep., 2002.

[14] D. Leffingwell, *Scaling Software Agility: Best Practices for Large Enterprises (The Agile Software Development Series)*. Addison-Wesley Professional, 2007.

[15] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.*, vol. 50, no. 9-10, pp. 833–859, Aug. 2008.

[16] M. Lindvall, D. Muthig, A. Dagnino, C. Wallin, M. Stupperich, D. Kiefer, J. May, and T. Kahkonen, "Agile software development in large organizations," *Computer*, vol. 37, no. 12, pp. 26–34, 2004.

[17] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *Journal of Systems and Software*, vol. 119, pp. 87–108, 2016.

[18] T. Dingsøyr and N. Moe, *Towards Principles of Large-Scale Agile Development*. Cham: Springer International Publishing, 2014, pp. 1–8.

[19] K. Petersen and C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices," *Empirical Softw. Engg.*, vol. 15, no. 6, pp. 654–693, Dec. 2010.

[20] E. Bjarnason, K. Wnuk, and B. Regnell, "A case study on benefits and side-effects of agile practices in large-scale requirements engineering," in *Proceedings of the 1st Workshop on Agile Requirements Engineering*, ser. AREW '11. New York, NY, USA: ACM, 2011, pp. 31–35.

[21] B. V. Craig Larman, "Scaling agile development," *CrossTalk*, pp. 8–12, 2013.

[22] D. Leffingwell, A. Yakyma, R. Knaster, D. Jemilo, and I. Oren, *SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering*. Addison-Wesley Professional, 2016.

[23] S. Ambler and M. Lines, *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM Press, 2012.

[24] V. Haren, *TOGAF Version 9.1*. Van Haren Publishing, 2011.

[25] P. A. Khosroshahi, M. Hauder, A. Schneider, and F. Matthes, "Enterprise architecture management pattern catalog version 2.0," Technische Universität München, Tech. Rep., 2015.

[26] E. Andersen, K. Grude, and T. Haug, *Goal directed project management: effective techniques and strategies*. Kogan Page Publishers, 2009.

[27] J. R. Turner, *People in project management*. Gower Publishing Company, 2003.

[28] S. Ambler and M. Lines, "Scaling agile software development: Disciplined agility at scale," Disciplined Agile Consortium, Tech. Rep., 2014.

[29] ——, "The disciplined agile process decision framework," in *International Conference on Software Quality*. Springer, 2016, pp. 3–14.

[30] ——, "Going beyond scrum disciplined agile delivery," Disciplined Agile Consortium, Tech. Rep., 2013.

[31] ——, "Scaling agile software development tactically: Disciplined agile delivery at scale," Disciplined Agile Consortium, Tech. Rep., 2016.

[32] "Disciplined agile 2.x: A process decision framework," http://www.disciplinedagiledelivery.com/, accessed: 2017-04-26.

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

[34] ——, "A lean non-functional requirements (nfr) framework: A common framework for governance, risk and compliance as well as traditional nfrs," AgilePath Corporation, Tech. Rep., 2017.

[35] A. B. Consortium, *DSDM Atern Handbook*. Agile Business Consortium, 2008.

[36] A. Craddock, K. Richards, D. Tudor, B. Roberts, and J. Godwin, "The dsdm agile project framework for scrum," DSDM Consortium, Tech. Rep., 2012.