Large-Scale Agile Frameworks: A Comparative Review

Fernando Almeida[1], Eduardo Espinheira[2]
[1]ISPGAYA & INESC TEC, Porto, Portugal
[2]Porto Business School, Porto, Portugal

Email: [1]almd@fe.up.pt

Abstract

This study aims to identify and systematically compare the main large-scale agile frameworks that companies can adopt to manage the work of large-scale and distributed teams. Through this, companies can more consciously perform a better-informed decision on the choice of the framework that best fits the practices and challenges of their organizations. This work employs a qualitative approach supported by an exploratory analysis that identifies and explores the processes of migration to a large-scale agile. In the first phase, fifteen assessment criteria for scaling agile are discussed. In a second phase, these criteria are used to perform a comparative analysis of six large-scale agile frameworks (i.e., DAD, LeSS, Nexus, SAFe, Scrum at Scale, and Spotify). The findings reveal there isn't a dominant large-scale agile framework in all dimensions. However, it is possible to identify frameworks like Nexus and Spotify that target smaller teams and offer low technical complexity. These frameworks easily accommodate changes, while there are other frameworks like SAFe and DAD that offer high levels of scalability but require more demanding and deep efforts in changing work processes in an organization.

Keywords: *Agile Development; Large-Scale; Project Management; Metrics; Scalability; Human Management*

1. Introduction

The globalization of the economy has forced organizations to face several challenges in the most wide-ranging business areas to achieve the agility and efficiency needed to remain competitive and adapt quickly to market changes [1]. As a result of this process, project managers have been challenged to adapt their processes to improve efficiency in increasingly agile environments, thus emerging alternatives to traditional methodologies for the management and development of projects.

The agile development was initiated through professionals connected to the software engineering area, aiming to minimize the risks associated with software development. The Manifesto for Agile Software Development was created in 2001 based on four fundamental values: (i) individuals and interactions over processes and tools; (ii) working software over comprehensive documentation; (iii) customer collaboration over contract negotiation; and (iv) responding to change over following a plan. Compared to the traditional model, agile methodologies work with short cycles or interactions, in which at the end of each step there is a deliverable product. Consequently, they provide faster changes that adapt to the current paradigm of technological evolution and high market competitiveness [2].

Agile practices have been highly successful in the corporate market, especially among small teams and projects [3]. However, its adoption in large organizational structures is often questioned, given the difficulties of managing independence among multiple teams, hierarchical pyramids that are inspired by non-agile models and the difficulties arising from a cultural heritage of the industrial era. This view is confirmed by [4], [5] who identified that the large-scale application and institutionalization of agile practices within companies that develop software is challenging since the adoption of agile practices in large teams and projects requires that agile principles be applied throughout the organization. Consequently, the introduction of agile practices challenges the existing practical structures and launches issues related to traditional roles, responsibilities, and expectations. In [6] it is argued that some decisions need to be made regarding the strategy of implementation of

agile methodologies, particularly in their expansion to other organizational areas and restructuring of business processes.

The success of agile methodologies in projects and small teams necessarily led to their adoption in new areas, with increasingly more companies applying agile practices in large-scale projects with teams involving hundreds of professionals, often geographically dispersed [7], [8]. As a consequence of this process, frameworks have emerged to manage large-scale agile projects, such as Large-Scale Scrum (LeSS), Disciplined Agile Delivery (DAD), Scale Agile Framework (SAFe), among others. These frameworks were developed considering a great variety of agile practices and covering multi and large teams in which the agile principles are applied throughout an organization. However, the choice of the most suitable and appropriate agile methodology for an organization is a problematic task. The results of the study carried out by [9] indicate that professionals in the engineering field pointed out the lack of an evaluation model to perform a comparative analysis among the various large-scale agile frameworks that allow guiding the practitioners in this choice. This situation has provoked little reflective and unsustainable decisions, in addition, to have paralyzed some transformation initiatives for large-scale agile. In this sense, this study intends to identify the large-scale agile frameworks that organizations can adopt and performs a comparative analysis between them to highlight the main characteristics that are essential in the process of choosing a large-scale agile framework.

The manuscript is organized as follows: Initially, a literature review in the field of large-scale agile frameworks is performed. Next, we present the organization of the study methodology in which the comparison criteria considered in the analysis of the frameworks are presented. After that, the results of this study are explored and discussed according to each dimension. Finally, the conclusions of this study are drawn.

2. Literature Review

Below is performed a literature review considering several frameworks for scaling agile.

2.1. DAD

The DAD was developed by Scott Ambler and Mark Lines in 2011 to fill the gaps in the Scrum processes [10]. DAD can be perceived as a hybrid approach that extends Scrum with other strategies derived from agile practices such as Extreme Programming (XP), Unified Process (UP), Kanban, among others. A key goal of DAD is to cover the entire delivery lifecycle from initial concept to delivery for operations and support. Table 1 presents the objectives of each DAD phase. The DAD life cycle consists of three phases in which a product is incrementally built.

Table 1. Objectives of each DAD phase

Phase	Objectives
Inception	Carrying out the project initiation activities. Identification of the project vision, stakeholders, initial requirements, forms of financing and project risks.
Construction	Production of a potentially consumable solution on an incremental basis. For this purpose, the project can be formed through a set of iterations or through a continuous and lean flow approach.
Transition	Ensure that the solution is ready to go into production and involve stakeholders in this process.

The roles suggested by DAD also assume a hybrid nature considering the size of the teams. It is important to recognize that roles are not people. Therefore, in DAD anyone can take on one or more roles and they may change over time. In [10] it is clarified that roles in DAD are divided into primary roles (e.g., team lead, product owner, architecture owner, team member, and stakeholder) that exist in all teams regardless of scale and secondary roles (e.g., specialist, integrator, domain expert, technical expert, and independent tester) that are only present in larger teams and for a certain period.

A DAD approach can scale from two perspectives: tactical agility and strategic agility. Tactical agility addresses the scalability process through a contextual application (e.g., team size,

geographic distribution, project complexity, etc.) of project objectives and practices that best fit it; strategic agility is more ambitious in advocating the implementation of lean and agile strategies throughout the organization [11].

2.2. LeSS

The LeSS methodology was conceived to apply Scrum to large projects, in multiple locations or offshore environments. LeSS specifies the organizational changes that must be developed in addition to those addressed in the traditional Scrum pattern and determines the creation of cross-functional teams through the elimination of traditional roles (e.g., project manager, team leader). In [12] it is reported that the use of agile methodologies enables the development of cross-functional teams, in which a combination of technical and functional skills is observed.

LeSS uses most of the practices and principles of other agile methodologies. The core objective is to always be very objective, simple, and transparent. Furthermore, the framework focuses on the entire product, customer focus, continuous improvement, lean thinking, among others. In LeSS, all sprints end at the same time and all Scrum teams work on the same product backlog [13]. Sprint Planning is also made up of two parts as in Scrum [13]: (i) the first part of the planning defines the objective and the product backlog items that will be included in the sprint; and (ii) the second part of the planning allows teams to set up their plan to develop the items and achieve the established objectives of the sprint.

LeSS is recommended for a maximum total of 8 teams with 8 members each. For larger teams there is also the LeSS Huge. Conceptually the Less Huge can be seen as the adoption of LeSS in an environment with multiple stacked LeSS structures. In the Less Huge, there is a change in roles with the emergence of a product owner per area, the emergence of requirement areas in the product backlog, and the execution of parallel LeSS sprint runs per requirements area [14].

2.3. Nexus

The Nexus framework was proposed by Ken Schwaber in 2015 and has the essential objective of facilitating the coordination of several Scrum teams (ideally between 3 and 9) in the development of a single product [15]. Nexus presents a set of rules, roles, and events very similar to what occurs in Scrum. The biggest difference between both frameworks is the focus that is given to the exploration of dependencies and synchronisms of the various Scrum teams. With this approach, Nexus seeks to increase the cohesion between teams with the ability to deliver a ready increment at the end of each sprint [15].

The Nexus integration team is a key element in this framework. This team is responsible for ensuring each delivery at the end of a sprint. To this end, they are responsible for solving the technical and non-technical problems that inhibit the Scrum teams from achieving their goals [15]. The Nexus integration team consists of two basic roles, as with the Scrum framework: (i) product owner; and (ii) scrum master. In addition, a third role has arisen which is the Nexus integration team. According to [16], its members are responsible for defining the application integration architecture and mentoring the Scrum teams. In this sense, this team should have the necessary cross-functional skills and resources to enable each Scrum team to progressively improve the state of the integrated increment. Nexus is also based on transparency, which turns possible to visualize the integrated state of increments of all artifacts.

2.4. SAFe

SAFe is an agile development framework that targets large-scale environments. The framework is divided into three segments: team, program, and portfolio [17]. Each level has its integration activities and processes. The SAFe incorporates lean and agile practices at all three levels, providing standards for team and program size, which can be employed at scale [17].

In the SAFe paradigm, an agile team remains similar to a typical Scrum team, with some minor variations. The Scrum team is now referred to as ScrumXP and there is still a Product Owner and a Scrum Master that can be shared between 2 to 3 teams. A ScrumXP team can be specialized, not necessarily cross-functional. Furthermore, ScrumXP teams should work in a cohesive and coordinated manner and should have the ability to design, build, and test their work. At the programme segment level, several roles have been created, namely: (i) product manager; (ii) system architect; (iii) release train engineer; and (iv) UX designer. In addition to these individual roles, the SAFe methodology proposes additional teams, such as: (i) business owner team; (ii) release management team; (iii) devops team; and (iv) system team.

The development functionalities in the SAFe is carried out synchronously by involving several teams in an Agile Release Train (ART). Moreover, SAFe considers that the teams regularly produce, every quarter, a Potentially Usable Increase (PSI). The ART stipulates that interactions should be structured and organized in timeboxes with fixed date and quality, but with variable scope [18]. The methodology proposes the organization of the ARTs in four two-week interactions, followed by a three-week interaction known as Hardening, Innovation and Planning phase (HIP). Hence, the teams dedicate themselves to ART, developing and synchronously launching the PSI at the same quarterly pace. At the portfolio level, the SAFe introduces concepts on investment themes and value flow for the alignment of ARTs at the program level. A value flow is seen as a long-lasting series of system definition, development and implementation process steps used to implement systems that provide a continuous flow of value to the business.

2.5. Scrum at Scale

Scrum at Scale is a framework that allows scaling the Scrum framework to multiple Scrum teams to address complex issues while delivering value to customers. Studies performed by [9], [12] indicate that the speed of teams and the volume of work produced decreases as the number of Scrum teams increases, mainly due to the work required to coordinate team dependency and duplicate work. In this sense, Scrum at Scale emerges as a means of structuring and coordinating the work of multiple teams and achieving linear scalability. Three principles for the Scrum at Scale framework can be identified: (i) light; (ii) easy to understand; and (iii) difficult to master. Scrum at Scale has two cycles: the Scrum Master cycle and the Product Owner cycle. Sutherland (2019) argues that the combination of these two cycles turns possible to create a framework that establishes the efforts of several teams in a single goal. Activities are coordinated by the Scrum of Scrums, which is built by the product owner (i.e., MetaScrum) and the various Scrum Masters of each team.

New roles and activities arise in the Scrum Master cycle. The Scrum of Scrums (SoS) is a Scrum team responsible for the incremental delivery of an integrated product at the end of each sprint. The same principles as the original Scrum are applied, such as team size, the existence of a Scrum Master and Product Owner. A new event called Scaled Daily Scrum (SDS) also appears that aims to identify impediments and discover dependencies among teams. One representative of each team (e.g., team's Scrum Master) must participate in the SDS. This model can be scaled to other levels of coordination with the emergence of the concept of Scrum of Scrum of Scrums (SoSoS). This multi-layered model requires steering and leadership that can be achieved through the Executive Action Team (EAT). The mission of EAT is to coordinate the various SoS and SoSoS teams and interact with other parts of the organization like top managers and financial managers. This organizational model enables the transparent identification of impediments that can be easily scaled in the organization and resolved on the same day. Moreover, this framework allows an organization to grow organically based on its needs and at a sustainable pace.

2.6. Spotify’s Agile Scaling Model

Spotify is currently one of the world's most popular music companies. Much of its commercial success is due to its work philosophy based on agile methodologies. Since the beginning, Spotify has adopted Scrum which worked well as the company had few collaborators. Meanwhile, the company's growth and the existence of development teams in several cities by different time zones forced the

company to develop internally a new method to scale the agile methodology and it provoked a mindset change in new employees. For this purpose, the company proposes a new model called the Spotify Tribe Engineering Model consisting of seven elements [19]: (i) squads; (ii) tribes; (iii) chapter; (iv) guild; (v) trio; (vi) alliance; and (vii) chief architect.

A Squad is similar to a Scrum Team. All Squad members must have the necessary skills and tools to design, develop and test the code. Each team has the autonomy to decide how to work (e.g., Scrum sprints, Kanban, TDD). It is also possible for each team to establish its agile work methodology at a micro-level [19].

A Tribe is made up of several teams that work on a given feature. Each tribe has a tribal lead that takes responsibility for creating an innovative and productive environment for their squads. In [19] it is argued that the squads of a tribe should share the same physical space and it is essential to promote collaboration between the squads. Furthermore, in [19] it is suggested that tribes should be composed of a maximum of 100 people to reduce the risks of increased bureaucracy and rigidity of work processes that arise in large teams. This recommendation is shared by [20] when stating that agile in large teams experience difficulties in coordination and cooperation between teams.

A Chapter and Guild have common objectives despite having different implementations. In both cases, the common goal is to keep the teams aligned and focused while ensuring the transparency of processes. However, a chapter is formed within the same tribe to discuss an area of specialization and its specific challenges. On the other hand, guilds are informal groups formed by people from different tribes but who have a common interest. Guilds are voluntary and are formed in the interest of the people. In [21] it is also pointed out that guilds are non-formal forms of knowledge sharing in large-scale agile organizations.

Finally, it is also important to look briefly at the other roles that arise in the Spotify model. Trio appears when in a tribe there is the tribe lead, product area, and design area. An alliance is a combination of three trios. Another key member is Chief Architect, who is responsible for defining the architecture of the technological solution and resolves the issues of system architecture dependency. In [22] it is argued that Chief Architects play an essential role in ensuring the system's integrity and evolvability, also functioning as a centralizing element that coaches the various agile teams.

2.7. Custom frameworks

The study conducted by [9] with global companies that adopt large-scale agile practices identified that 7 in 13 of the companies (e.g., Accenture, Dell, Intel) use custom frameworks developed by the companies themselves. These models are inspired by several agile methodologies such as Scrum, Kanban or TDD. Many of these developments were not created from scratch but emerged due to failed initiatives in the adoption of previous models of large-scale agile.

One of the reasons that emerges for companies to develop their custom framework is the goal of creating a framework that effectively works in practice and that will add value and reduce delivery times [9]. Another reason is the specificity of each company, which hinders the adoption of frameworks with very strict rules and that can be an obstacle to their adoption by organizations [9]. In fact, a model that results in one organization may not produce the same results in another type of organizational culture. Moreover, the studies carried out by [23] identified that specific organizational culture factors correlate with the effective use of an agile method. Another issue that stimulates the creation of custom frameworks is the compliance processes that need to be developed by following national and international regulations [24]. Indeed, there are markets (e.g., health sector) that require more robust and extensive processes (e.g., the testing process and quality assurance).

3. Method

This study uses a qualitative approach supported by an exploratory analysis considering secondary sources that explore the processes of migration to large-scale agile. According to [25], the exploratory study enhances the ideas or discovers insights, being generally employed when there is little knowledge about the topic to be addressed. Furthermore, qualitative research emphasis on

situational specifics and contributes to a good description of processes [26]. In fact, the migration process to large-scale agile is still in an emerging phase, and there are mostly published studies that explore the challenges and good practices of this migration in various organizations through case studies. In these studies, the existence of two approaches becomes evident: the analysis of migration challenges for large-scale [4], [5] and the exploration of the specific challenges of using large-scale frameworks like LeSS or SAFe [13], [27]. However, there is a need to have a study that comparatively evaluates these frameworks according to multiple criteria for scaling agile.

The data collection process in this study was performed in two stages. The first step covered the identification of large-scale agile frameworks. In total six large-scale agile frameworks were identified based on the literature review process. Furthermore, it was discovered the existence of custom frameworks developed by the companies and that meet the particularities of each company and sector of activity. In the second stage, the authors sought to identify criteria that would allow a comparative analysis of the frameworks. For this purpose, the challenges and success factors for large-scale agile transformations identified by [5], [9] were considered. Added to these factors were included the expected benefits from scaling agile identified by [27], [28], which allow meeting the benefits expected by companies when they perform a migration process for large-scale agile.

Table 2 maps the criteria used in the process of benchmarking frameworks. It can be observed that most of the criteria are common to all four authors. However, there are some less known criteria such as waste elimination and learning ability that may be relevant in the process of comparative analysis of large-scale agile frameworks.

Table 2. Assessment criteria for scaling agile

Criteria	Ref [5]	Ref [28]	Ref [27]	Ref [9]
Accommodate changes	X	X	X	X
Continuous improvement	X	X	X	
Control level	X	X	X	X
Coverage	X		X	X
Customer involvement	X	X	X	X
Ease to use	X	X	X	X
Flexibility	X	X		X
Geographically distributed	X		X	
Learning ability				X
Scalability	X	X	X	X
Team size	X	X	X	X
Technical complexity	X	X	X	X
Time to market		X	X	X
Transparency	X	X	X	X
Waste elimination		X	X	

4. Results and Discussion

Table 3 performs a comparative analysis of the characteristics and potentialities offered by each large-scale agile framework. For this purpose, a nominal scale composed of three gradual indicators (e.g. low, moderate, and high) was established as suggested by [29] and that can be applied both in the engineering and social sciences fields. In the scope of this study, the Spotify framework was included in the comparative process of analysis of large-scale agile frameworks, although its model is not unanimously considered a large-scale framework. This view is shared by [30] who points out that the Spotify model was applied in the cultural and contextual background of the Spotify company, not aiming to define itself as a framework for scaling agile. However, and despite this, the study conducted by [9] finds several organizations in the financial and public sectors adopting this approach to solve the challenges of adopting agile on a large scale. In this sense, this framework was considered in this study, but its column in Table 3 is shaded in grey to highlight this situation.

Table 3. Comparative analysis of large-scale agile frameworks

Criteria	DAD	LeSS	Nexus	SAFe	Scrum at Scale	Spotify
Accommodate changes	Moderate	Low	High	Low	Low	High
Continuous improvement	Low	High	High	Low	Low	High
Control level	Moderate	High	High	High	Moderate	Moderate
Coverage	High	Moderate	Moderate	High	Moderate	Moderate
Customer involvement	Moderate	High	Moderate	High	Moderate	Moderate
Ease to use	Low	Low	Low	Moderate	Moderate	Moderate
Flexibility	Low	Low	Low	Low	Moderate	Moderate
Geographically distributed	Moderate	Moderate	Moderate	High	Low	Moderate
Learning ability	High	High	Moderate	Low	Low	High
Scalability	High	Moderate	Low	High	Low	Moderate
Team size	Moderate	Moderate	Low	High	Low	Low
Technical complexity	Moderate	Moderate	Low	High	Low	Low
Time to market	Moderate	Moderate	Moderate	Moderate	Moderate	High
Transparency	High	High	High	High	Moderate	High
Waste elimination	Low	Low	Low	Low	Moderate	Low

4.1. Accommodate changes

According to [17], requirements may change according to several conditions, such as forgetting a requirement, changing the customer's opinion, marketplace changes, political or legislative issues. In this sense, large-scale agile frameworks have to offer the possibility to accommodate changes. Nexus and Spotify are the frameworks that best accommodate changes in requirements by minimizing dependencies and process control. Both frameworks address short term challenges, which facilitate the inclusion or reformulation of a requirement. On the other hand, in LeSS, SAFe and Scrum at Scale the proposed organic structures are relevant to ensure a high follow-up of the processes, but they become little reactive to the changes that may arise in the requirements. This view is confirmed by [6] when highlighting that the challenges of incorporating a large-scale agile framework arise mainly at the organizational level, which slows down the change process. In this sense, the process of change management is not only at the technical level but also at the organizational level, such as budgeting and metrics [31].

4.2. Continuous improvement

Continuous improvement is an adopted practice that aims to improve results by making them more efficient and effective, whether in products, services, or processes. According to [32], continuous improvement requires continuity, it is cultural to focus not only on company processes, but also on their cultural dimension and the benefits offered should be comprehensive and involve all areas. The implementation of a continuous improvement methodology must evidence incremental (e.g., occurring slowly, and with small increments, over time) or immediately results (e.g., significant and sudden change).

In addition to the two previous frameworks (i.e., Nexus and Spotify) that offer high capacity for accommodate changes and continuous improvement, LeSS also stands out. In LeSS, the change initiative is continuous through experimentation and improvement. These three practices contribute to eliminate the traditional project/program manager positions and allow the deployment process to be significantly lower. On the opposite side, frameworks like SAFe are too prescriptive, which hinder the process of continuous improvement in short cycles.

4.3. Control level

All the analyzed frameworks contribute to reach an acceptable level of process control, especially LeSS, Nexus, and SAFe that ensure an efficient control of the sprints of each increment. In these frameworks, a top-down control model with a bottom-up execution strategy is adopted. One of the essential aspects of these frameworks is the contribution to an increase in project predictability and risk control. According to [5], in large-scale development, the project risks necessarily increase significantly, therefore it is relevant to predict, anticipate and mitigate problems and defects that become more expensive as the development process evolves.

It is also important to analyze the trade-off between a high control level and the autonomy of a team. In the vision of [9], large-scale agile frameworks can aggravate this problem by imposing more restrictions and rigidity in a process that is necessarily intended to be agile and dynamic. In this sense, the level of control offered by a framework should be analyzed in conjunction with other factors like scalability, autonomy, and flexibility.

4.4. Coverage

An organization that works with dozens or hundreds of agile teams also needs to be agile in its other organizational areas. In [33] it is pointed out that the agile units of a newly converted company may suffer from bureaucratic procedures or lack of collaboration between the operation, management and innovation teams. In this sense, organizational changes should be implemented to ensure that the agile teams are compatible with the functions that do not operate in this way. A large-scale agile framework must necessarily cover the entire enterprise, and not exclusively the operational processes.

In terms of coverage, SAFe and DAD frameworks stand out. The SAFe covers the entire enterprise areas and is structured in three dimensions (e.g., team level, program level, and portfolio level; while the DAD includes four levels (e.g., Team level, DevOps, IT, Enterprise).

4.5. Customer involvement

The principles of agile management argue that customer satisfaction is a higher priority than the processes used during the project. The active involvement of the client throughout the project gives agile teams the chance to understand the client's vision and develop a reliable and authentic relationship with the team [34]. On a large scale, these challenges gain greater importance since the impact of developing something that is not in line with the customer's needs is also greater. In this dimension, two frameworks stand out: (i) in the SAFe, constant feedback from customers allows for the inclusion of improvements throughout the process; and (ii) LeSS offers a customer-centric and feature-oriented approach in which the teams engage directly with the customer, allowing them to define priorities and a long-term vision of the product.

Both the SAFe and LeSS frameworks advocate that teams should work closely with external and internal customers. In fact, in [9] it is emphasized this approach reduces the layers of control and approval, accelerating the work and increasing the motivation of teams. Consequently, this approach encourages the emergence of a bottom-up approach to innovation, in which the responsibility for innovation lies with those who are closest to customers.

4.6. Ease to use

The empirical study carried out by [9] among European companies that use large-scale agile frameworks identified that one of the main difficulties experienced by companies is the trouble in

using these frameworks in practice. This study confirmed those findings, but additionally it turned possible to identify three frameworks whose adoption process is less complex: (i) SAFe; (ii) Scrum at Scale; and (iii) Spotify. The adoption of the SAFe provides a global vision of the projects and does not require a high restructuring of the processes in the company. Moreover, SAFe is a framework that is well documented. The Scrum at Scale is a simple and effective framework that reduces and avoids the introduction of extra complexity. Consequently, the productivity per team does not decrease as more teams are created. Finally, the Spotify framework effectively addresses short-term challenges and responds to changes quickly.

4.7. Flexibility

As might be expected, in larger organizations it becomes more difficult to implement a transformation that involves changes throughout the value chain, from processes to behaviors and cultures. It is important to recognize that agile methodologies are based on principles of flexibility, close collaboration, small autonomous units, and clear communication. However, scaling it up in organizations that require several different units involved in individual decisions is a challenge. Furthermore, in [4] it is argued that organizations need to recognize that the use of agile methodologies is not simply a change in the way a product is delivered, but rather a behavior change that needs to be evident in delivery teams and other peripheral units (e.g., finance, human resources, etc.).

Two frameworks stand out in terms of flexibility: Scrum at Scale, and Spotify. Scrum at Scale offers fluid communication between various scrum teams through SoS meetings. Dependencies are thus managed by this framework. In the Spotify framework, the teams are autonomous and self-managed with minimal control. This approach minimizes dependencies within teams.

4.8. Geographically distributed

The scheduling of teams according to market conditions is key for companies. One of the market trends is the adoption of distributed teams in which companies look for the best professionals in the industry regardless of their geographical location. Although this model is not new, the potential offered by information technologies makes working in a remote environment easier to implement. In [35] it is summarized some of the main advantages of this approach, namely the increase in productivity, the balance between personal and professional life and the attraction of talent on a global scale. However, there are also some challenges, namely the difficulty of coordinating across different time zones, the centralized monitoring of the development process, communication difficulties, the sharing of informal knowledge and the cultural differences that may arise [7].

The high level of control offered by the SAFe and the offer of a decentralized decision-making make process turns this framework the most suitable for working in large organizations with geographically distributed teams. Furthermore, the SAFe is a framework that encourages the intrinsic motivation of highly qualified workers, who naturally arise in environments with geographically distributed teams. The findings obtained by [18] reveal the application of the SAFe in a large European company with approximately 115,000 employees and in which the SAFe was used in the migration processes from waterfall to agile environment turned the company more efficient by reducing the average project development time by approximately three months.

4.9. Learning ability

One of the properties of agile methods is their high flexibility based on iterative and incremental cycles. Throughout this process, in [12] it is argued there should be a commitment of people to constant learning rather than following a strict agile framework or method. In this way, teaching teams how to learn throughout the development cycle is an element that will contribute to increasing the maturity of teams and the delivery of more reliable products.

Three frameworks stand out from this perspective. DAD offers a learning-oriented hybrid agile approach in which the sharing of knowledge with other teams is encouraged. Furthermore, one of the fundamental principles of DAD is to allow the best choice through a goal-oriented approach and the support of multiple life cycles. LeSS is another framework that offers high learning conditions and consistent growth. This situation is achieved through frequent coordination between teams and the existence of frequent retrospective sessions that contribute to the continuous improvement of the project. Finally, Spotify is also another framework in which team learning is strongly promoted through continuous integration processes.

4.10. Scalability

Scalability can be seen as the ability to adjust or adapt the framework to work needs with a larger volume of people and teams in parallel. In [36] it is stated that in an agile-scale framework scalability should be considered both from a vertical and horizontal perspective. From a vertical perspective, agile transformation is shared by several business units that execute their own agile initiatives simultaneously or sequentially. In the end, the resulting work of multiple teams is integrated into a single product. From a horizontal perspective, a specific agile structure is adopted and other business units progressively replicate the same model.

SAFe is a framework that presents high levels of scalability by facilitating the work with multiple teams, in which the organizational structure, responsibilities, and roles are well defined. Another framework that presents a high level of scalability is DAD by supporting diverse delivery lifecycles (e.g., agile lifecycle, lean lifecycle, program lifecycle) and by not providing a single life cycle, which turns possible to be adjusted through context.

4.11. Team size

With the increasing size and complexity of projects, the techniques used to manage them are no longer effective. The large-scale agile frameworks must also meet this challenge. For simple small-scale projects, Scrum at Scale, Nexus and Spotify frameworks promote integration and continuous improvement and high transparency of projects. On the contrary, in projects that involve large teams and need an agile transformation, the most appropriate solutions are SAFe and Less - Huge. The SAFe is particularly designed for large companies facilitating the work with several teams. This framework also promotes the sharing of strategy and architecture between the development and management teams.

4.12. Technical complexity

Low complexity is a very important point for agile development. In [13] it is stated the simpler and less confusing the processes, the easier it is for the team and the client to understand. Furthermore, the greater the understanding, the more fluid the product development process will be. The frameworks that present lower levels of complexity are also those aimed at small and medium-sized teams (e.g., Nexus, Scrum at Scale, and Spotify). On the opposite side, the SAFe defends a prescriptive model with heavy processes that require an agile transformation of the organization. In addition to the weight of the process, the SAFe also adds layers of management throughout the development process.

4.13. Time to market

One of the objectives of agile methodologies is to offer greater agility and speed in the delivery of projects. In this sense, large-scale agile frameworks should contribute to the elimination of waste, increase the speed of delivery, respond more quickly to market changes and improve development processes. However, being agile does not mean delivering the entire project faster. The focus should be on delivering faster value to the customer since the deliveries are partial and incremental.

The Spotify framework is one that offers shorter incremental delivery cycles. This improves delivery speed through continuous integration processes and integration of code into a delivery patch. Furthermore, based on customer feedback, it is possible to improve a feature or switch to an older feature, if needed. As a result, responses to changes are achieved much more quickly.

4.14. Transparency

Transparency is a core value for Agile to work on a large scale. Transparency gives a company the ability to evaluate and track work progress and to inspect and adapt its products and processes. Large-scale agile frameworks offer high standards of transparency by providing a view of the entire product that ensures transparency in the work carried out by teams. However, from a practical point of view, the use of large-scale agile frameworks by companies does not always provide a correct and complete view of the processes. The findings identified by [9] report the existence of an excessive tendency of companies to evaluate their agile transformation level through metrics that reveal the adherence level to a framework (e.g., number of tribes, number of user stories in a sprint, etc.). These metrics give inaccurate indications about the impact of using these frameworks on key performance metrics like value delivery, quality, productivity, or sustainability.

4.15. Waste elimination

The waste is an intrinsic element of any system and gradually becomes more visible in complex systems. As an artifact is built it increases the likelihood of waste generation and the greater the need to spend time and effort to correct errors and impediments [37]. The study conducted by [10] concluded that the three main causes for waste generation are the inclusion of unsolicited functionality, project churn (e.g., projects and tasks switching) and crossing organizational boundaries mainly between the development team and stakeholders. One of the ways to reduce the amount of waste is to have self-organized teams that operate around the work that is trying to accomplish.

Most large-scale frameworks have difficulties in reducing the waste generated at each stage of development, mainly due to the rigidity of processes and the complexity of projects. Moreover, projects in which there is great instability of requirements tend to generate more waste. Scrum at Scale is the one that best responds to this challenge through the modular structure and focuses on the management of the premises and through the incentive to the creation of autonomous and multifunctional teams. However, the answer to this challenge becomes more complex when the value flows intersect several teams.

5. Conclussion

Agile development is an incremental, iterative, and collaborative approach to project development that contributes to the development of high quality and low-cost products. Initially, agile practices were launched to support small teams that share the same physical space but can be adapted to suit the more complex environment. The large-scale agile frameworks that have emerged in the market meanwhile propose an agile and disciplined delivery lifecycle that scales agile building strategies to handle the entire delivery process, from the beginning of the project, to development, and production. In this sense, these frameworks explicitly deal with the complexities faced by large teams, namely in terms of control level, coverage, flexibility, scalability, transparency, waste elimination, among others.

Today's companies find it very difficult to choose a large-scale agile framework that fits their business specificities. In fact, each organization faces a combination of different factors and therefore needs a process, a team structure, and a toolset environment tailored to their unique situation. In this sense, this study seeks to give undeniable theoretical and practical contributions by defining a structure that allows comparing the various frameworks, thus also supporting companies in choosing the best framework that fits the practices and challenges of their organizations. The findings did not reveal a dominant framework that is better all dimensions, but it is possible to consistently identify the

existence of frameworks that stand out in a set of dimensions. Furthermore, the findings indicate that the main challenge faced by organizations in the process of agile scaling is to understand internally their organizational culture and not try to impose a large-scale agile framework based on an organizational culture distinct from the mindset of their organization. In this sense, a framework that presents good results in one organization can be a failure in another.

As future work, it is intended to build a decision support system that can be a technological application to support companies in choosing a large-scale agile framework. In this sense, it is intended to adopt a multi-attribute method (e.g., AHP, MAUT) that helps organizations to choose and understand their choice, in which the importance of each criterion is established by the company.

References

[1] M. Dülgerler, "Making better, more responsive organizations", In *Proceedings of the PMI® Global Congress 2015 - EMEA*, London, England, 2015.

[2] S. Alsaqqa, S. Sawalha, and H. Abdel-Nabi, "Agile Software Development: Methodologies and Trend", *International Journal of Interactive Mobile Technologies*, vol. 14, no. 11, pp. 246-270, 2020.

[3] S. Lee, and H. S. Yong, "Agile Software Development Framework in a Small Project Environment", *Journal of Information Processing Systems (JIPS)*, vol. 9, no. 1, pp. 69-88, 2013.

[4] T. Dingsøyr, and N. Moe, "Towards Principles of Large-Scale Agile Development: A Summary of the workshop at XP2014 and a revised research agenda", In *Proceedings of the 15th International Conference on Agile Software Development*, Rome, Italy, pp. 1-9, 2014.

[5] 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.

[6] M. Paasivaara, B. Behm, C. Lassenius, and M. Hallikainen, "Large-scale agile transformation at Ericsson: a case study", *Empirical Software Engineering*, vol. 23, no. 5, pp. 2550-2596, 2018.

[7] F. Almeida, E. Miranda, and J. Falcão, "Challenges and facilitators practices for knowledge management in large-scale scrum teams", Journal of Information Technology Case and Application Research, vol. 21, no. 2, pp. 90-102, 2019.

[8] D. Marjanovic, M. Storga, S. Skec, N. Bojcetic, and N. Pavkovic, "Agile beyond software – a study of a large scale agile initiative", In *Proceedings of the Design 2018 15th International Design Conference*, Dubrovnik, Croatia, pp. 2055-2062, 2018.

[9] K. Conboy, and N. Carroll, "Implementing Large-Scale Agile Frameworks: Challenges and Recommendations", *IEEE Software*, vol. 36, no. 2, pp. 44-50, 2019.

[10] S. Ambler, and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*, IBM Press, New York, 2012.

[11] G. Schuh, E. Rebentisch, C. Dölle, C. Mattern, G. Volevach, and A. Menges, "Defining Scaling Strategies for the Improvement of Agility Performance in Product Development Projects", *Procedia CIRP*, vol. 70, pp. 29-34, 2018.

[12] M. Kalenda, P. Hyna, and B. Rossi, "Scaling agile in large organizations: Practices, challenges, and success factors", *Journal of Software: Evolution and Process*, vol. 30, no. 10, pp. 1-25, 2018.

[13] C. Larman, and B. Vodde, *Large-Scale Scrum: More with LeSS*, Addison-Wesley, Boston, Massachusetts, 2016.

[14] G. Blokdyk, *Large-Scale Scrum*, 5STARCooks, Plano, Tx, 2018.

[15] K. Bittner, *Nexus Framework for Scaling Scrum: The Continuously Delivering an Integrated Product with Multiple Scrum Teams*, Addison-Wesley Professional, Boston, Massachusetts, 2017.

[16] P. Firat, and E. Can, "Two case studies to explore pros/cons and to assess applicability of Nexus maturity model", *African Journal of Business Management*, vol. 12, no. 6, pp. 154-160,

2018.

[17] D. Leffinhwell, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*, Addison-Wesley Professional, Boston, Massachusetts, 2011.

[18] O. Turetken, I. Stojanov, and J. Trienekens, "Assessing the adoption level of scaled agile development: a maturity model for scaled agile framework", *Journal of Software: Evolution and Process*, vol. 29, no. 6, e1796, 2017.

[19] H. Kniberg, and A. Ivarsson, "Scaling Agile @Spotify with Tribes, Squads, Chapters & Guilds", 2012, available at: https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf

[20] L. Ng, " Why the Agile Method Often Fails in Big Companies", 2019, available at: https://medium.com/swlh/agilefall-the-trap-that-software-development-often-falls-into-e092165c6c28

[21] D. Smite, N. Moe, G. Levinta, and M. Floryan, "Spotify Guilds: How to Succeed With Knowledge Sharing in Large-Scale Agile Organizations", *IEEE Software*, vol. 36, pp. 51-57, 2019.

[22] G. Hohpe, I. Ozkaya, U. Zdun, and O. Zimmermann, "The Software Architect's Role in the Digital Age", *IEEE Software*, vol. 33, no. 6, pp. 30-39, 2016.

[23] D. Strode, S. Huff, and A. Tretiakov, "The Impact of Organizational Culture on Agile Method Use", In *Proceedings of the 42th International Conference on Systems Science*, Waikoloa, Hawaii, pp. 1-9, 2009.

[24] M. Kostic, " Challenges of agile practices implementation in the medical device software development methodologies", *European Project Management Journal*, vol. 7, no. 2, pp. 36-44, 2017.

[25] A. Queirós, D. Faria, and F. Almeida, "Strengths and Limitations of Qualitative and Quantitative Research Methods", *European Journal of Education Studies*, vol. 3, no. 9, pp. 369-387, 2017.

[26] R. Gephart, " Qualitative Research and the Academy of Management Journal", *Academy of Management Journal*, vol 47, no. 4, pp. 454-462, 2004.

[27] A. Putta, M. Paasivaara, and C. Lassenius, "Benefits and Challenges of Adopting the Scaled Agile Framework (SAFe): Preliminary Results from a Multivocal Literature Review", In: Kuhrmann M. et al. (eds) *Product-Focused Software Process Improvement*, PROFES 2018, Lecture Notes in Computer Science, vol. 11271, Springer, Cham, 2018.

[28] U. Eklund, and C. Berger, "Scaling agile development in mechatronic organizations - a comparative case study", In *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, Buenos Aires, Argentina, pp. 137-182, 2017.

[29] R. Johnson, and G. Morgan, *Survey Scales: A Guide to Development, Analysis, and Reporting*, The Guildford Press, New York, 2016.

[30] A. Mersino, "There Is No Spotify Model for Scaling Agile", 2019, available at: https://vitalitychicago.com/blog/there-is-no-spotify-model-for-scaling-agile/

[31] J. Karlsson, " Principles of Good Large-Scale Agile", 2019, available at: https://thenewstack.io/principles-of-good-large-scale-agile/

[32] A. Poth, S. Sasabe, A. Mas, and A. L. Mesquida, " Lean and agile software process improvement in traditional and agile environments", J*ournal of Software: Evolution and Process*, vol. 37, no. 1, pp. 1-11, 2019.

[33] D. Rigby, J. Sutherland, and A. Noble, "Agile at Scale", *Harvard Business Review*, vol. May-June, 2018, available at: https://hbr.org/2018/05/agile-at-scale

[34] S. Bambauer-Sachse, and T. Helbling, "Customer satisfaction with business services: is agile better?", *Journal of Business & Industrial Marketing*, In Press, 2021.

[35] E. F. Turesky, C. D. Smith, and T. K. Turesky, "A call to action for virtual team leaders: practitioner perspectives on trust, conflict and the need for organizational support", *Organization Management Journa*l, vol. 17, no. 4/5, pp. 185-206, 2020.

[36] P. Maddaloni, "Agile at Scale: Comparing the Popular Scaling Frameworks", 2019, available at: https://www.6kites.com/comparison-scaling-agile-frameworks/

[37] M. Wiesche, "Interruptions in Agile Software Development Teams", P*roject Management Journal*, vol. 52, no. 2, pp. 210-222, 2021.