Scaling Scrum with a Customized Nexus Framework: A Report From a Joint Industry-Academia Research Project[1]

Andrzej Joskowski, MSc
Intel Technology Poland
Słowackiego 173, 80-298 Gdańsk, Poland

Adam Przybyłek, PhD
Gdansk University of Technology; Faculty of Electronics, Telecommunications and Informatics
Narutowicza 11/12, 80-233 Gdańsk, Poland
orcid.org/0000-0002-8231-709X

Bartosz Marcinkowski, PhD; DSc (corresponding author)
University of Gdansk; Department of Business Informatics
Piaskowa 9, 81-864 Sopot, Poland
bartosz.marcinkowski@ug.edu.pl
orcid.org/0000-0002-7230-2978

Data availability statement: All the relevant data are included in the body of the article
Funding statement: No external funding was used to carry out the study
Conflict of interest disclosure: Andrzej Joskowski reports a relationship with Intel Technology Poland that includes employment. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
Ethics approval statement: Not applicable.
Patient consent statement: Not applicable.
Permission to reproduce material from other sources: Not applicable.
Clinical trial registration: Not applicable.

Scaling Scrum with a Customized Nexus Framework: A Report From a Joint Industry-Academia Research Project

Abstract

Despite a wide range of scaling frameworks available, large-scale agile transformations are not straightforward undertakings. Few organizations have structures in place that fit the predefined workflows – while once one applies an off-the-shelf framework outside of its prescribed process, guidance quickly runs out. In this paper, we demonstrate how to instantiate a method configuration process using a lightweight experimental approach embedded in Action Research cycles. The proposed approach was developed to assist practitioners working on a multiple-team project at Intel Technology Poland to find the right practices to continue their Nexus-based transformation and integrate their in-house method into the already established company structures, processes, and routines. In particular, it enabled identifying a series of challenges with scaled practices and coping with those. The challenges ranged from logistical problems, through poor availability of the Product Owner, to lackluster knowledge transfer and a wide array of communication/coordination issues at meetings. The study broadens the current body of knowledge within technology management and the scaled agile method tailoring domain. It indicates potential corrective actions that may be taken advantage of by entities that are not inclined, due to organizational constraints, to directly implement off-the-shelf frameworks.

Keywords

Large-scale Agile; Method Configuration; Software Process Improvement; Agile Process Tailoring; Agile Transformation; Organizational Change; Action Research

1 Introduction

Agile methods have revolutionized the way software projects are managed. Their effectiveness across small, single-team projects[9] has inspired companies to increasingly adopt them within large-scale endeavors with multiple development teams.[65] To support this process, custodians of existing agile methods and consultants have proposed different frameworks which are claimed to provide ready-made solutions to scaling agile.[28] Nonetheless, as the vast majority of software development projects are unique, they cannot be properly supported by any off-the-shelf framework implemented in its original, textbook format.[8, 13, 21, 30, 35, 41, 54, 66, 69, 73] Besides, not all organizations are ready to fully implement any framework in a one-time shift regarding organizational, cultural, and technical aspects,[33, 73] whereas there are no gradual approaches or guidelines for large-scale agile transformation.[10, 42] Indeed, previous studies have reported numerous challenges while adopting such frameworks,[8, 21, 27] including a mismatch between framework and organization,[21, 30, 73] controversies within the framework,[6, 29, 46] complex organization setup,[10, 46] changes in management structure,[10, 21, 29, 31, 32] fluctuating company's policies,[10, 29, 31] and change resistance.[10, 29, 46] Not surprisingly, it has been acknowledged that agile methods often need to be tailored to accommodate specific situations[21, 30, 35, 33, 34, 36, 69, 71] or issues (e.g. risk management).[56]

Tailoring software engineering methods can be classified into one of two approaches – namely, situational method engineering and method configuration[13, 23] (some researchers consider method configuration as a particular form of situational method engineering).[24, 25] The former is a meta-method process, where a new method is constructed or "engineered" from the ground up using existing "method fragments" relevant to the situation at hand.[35, 13] These fragments are selected and integrated to take advantage of the strengths of different methods.[33, 23] The latter, on the other hand, always takes one particular method as a starting

point that is configured to the specific development situation.[34, 23, 24] Nonetheless, the base method may need to be enhanced with additional fragments from other methods, while some of its original fragments may need to be removed.[24] As far as agile software development is concerned, method fragments are known as agile practices.[55, 53]

Regardless of which specific approach is adopted, tailoring is not trivial and poses serious challenges for practitioners[33] even in the context of agile methods[36, 46, 47, 54] despite the fact that "*anything labeled as agile should itself be flexible and amenable to tailoring*".[35] Firstly, practitioners usually do not know how to carry out method tailoring in a disciplined manner, and how to evaluate each method fragment before deciding whether to adopt it or remove it,[13, 33, 35] whereas unstructured, ad hoc tailoring results in projects being subjected to what the practitioners called the "haphazard" or "patchy" use of a method.[35] Secondly, practitioners should be familiar with a broad range of methods to allow the effective combinations or substitution of method fragments according to the project specificity.[13, 33, 35] However, comprehensive knowledge of even one full method is rare among developers.[35, 71] Thirdly, very little is known about the degree to which individual framework practices are interconnected, interdependent, and synergistic as well as the implications of removing certain practices or plugging in practices from other frameworks.[35, 25]

Taking into account the aforementioned considerations, organizations usually need help choosing the right combination of practices for their environment.[33] Indeed, in this paper, we report on a joint research project between Intel Technology Poland and academia, in which we helped Scrum teams to find the right agile practices to carry out a stepwise large-scale agile transformation based on a tailored Nexus framework and integrate the practices into the already established company structures, processes, and routines. The main deliverable of our work is a lightweight Action Research-based approach to guide academics and practitioners through the iterative and incremental process of selecting and adopting practices to configure the framework in place to fit the project characteristics.

2 Background and Related Work

2.1 Large-scale Agile Transformations

Transforming an organization towards large-scale agile is far from a mechanical carryover of agile tools and practices to a correspondingly larger number of collaborating teams. Julian et al. identified two general means of adopting agile in an organization: the big bang and gradual adoption.[47] With a big bang, teams perform a holistic adoption of a framework across the whole organization. After teams develop expertise in using new practices, they focus on continuous improvement, iteratively tailoring the practices to better fit their needs. In contrast, teams following a gradual approach take on a few new practices at a time and integrate them into the existing ones. Rolland et al. point out that although agile principles tend to be considered unassailable or even sacrosanct by many researchers, those, in fact, cannot be always directly applied while scaling.[5] They challenged straightforward extrapolation practices such as Scrum → Scrum-of-Scrums, emphasizing the fact that from a certain point, knowledge boundaries come into play across different actors that need to be addressed with dedicated workflows and infrastructures. Schnitter and Mackert, who made early attempts to scale Scrum to meet the needs of a large team at SAP AG, came to the conclusion that 130 professionals constitute a natural limit, beyond which extraordinary company-wide measures are necessary.[14] They anticipated that the scaled approach required about one year to become fully operational. And the learning curve is unlikely to go away – without mentor support in place, symptoms of erosion emerge.

Therefore, challenges and success factors are at the heart of many studies. A case study performed by Saeeda et al. highlights that many of such challenges allude to testing highly elaborate IT artifacts, the delivery of which involves numerous teams.[68] That said,

transformation-related issues are highly diverse. As a matter of fact, Abrar et al. listed as many as fifteen de-motivators for scaling agile effectively.[43] Companies strive for completing the transformation process too fast, their employees are reluctant to change, issues with assuring quality emerge, and legacy non-agile organizational artifacts prove to be hard to integrate.[6] Organizations leave training out of transformation's scope, maintain as-usual workloads despite the transformation process itself temporarily yielding new workflows, and keep a number of old commitments that under the new reality should be discontinued.[8] Individual members and their teams fail to keep track of other teams' work, have trouble estimating how their work impacts their extra-team counterparts and manage interdependencies that appear out of the blue when the tasks are closed, and simply get confused about some roles and responsibilities.[10, 44, 57] Such coordination and backlog/requirement management weaknesses and shortcomings in multi-team settings are approached with a wide spectrum of meetings[15] of both formal and informal nature.[11] Practitioners attempt to devise and field-test specialized "learning by doing" self-service kits that enhance skills and mindset as well as position expert agile practitioners as prosumers to counterbalance agile competency gaps.[3, 62] As agile methods might be classified as simple by design, but difficult to implement well, focusing on practical training in this regard cannot be overestimated. Hence, the agile and lean-related experience came up prominently among the key success factors for a successful transition to scaled agile alongside such factors as actual support from the management support, aligning values and viewpoints, as well as enhancing agile culture throughout an entire company.[6] Carroll et al. stress that the transformation process itself ought to be supported by dedicated resources and demonstrate the added value of agile coaches, sourced externally for transformation.[1]

As the number of teams expands, a natural dilemma arises regarding a product owner's organizational setup[12] in the overall endeavor enabling him/her to cope with the numerous potential challenges[70] and the scope of his/her support. Whereas conveying the vision and outlying team priorities is the paramount consideration here, the extent of delegation of authority and the closeness of collaboration with individual teams might be considered basic differentiators.[2] Whereas the generic product owner role may feature an exceptionally wide range of activities and responsibilities, large-scale agile environments in practice make such a spectrum ineffective.[16] Hence, having the role mapped to multiple closely related actors might be a viable option. Ultimately, the product owner's role is likely to evolve as the transition gains in maturity.

To mitigate the most common mistakes, some parties decide to invest heavily in frameworks for scaling agile. Theobald et al. went beyond commonly recognized frameworks, confronting underlying practices and coming up with a classification.[17] Whereas an informed decision regarding which existing framework potentially constitutes the best fit for a company helps to avoid some of the pitfalls, some barriers end up getting inherited. And some new, specific ones might emerge. For instance, Putta et al. when scrutinizing the implementation of the SAFe framework within a large financial corporate body encountered a series of challenges, two of which could potentially derail any SAFe-centered organizational transformation: (1) the management of traditional companies might be unwilling to embark on a path of a far-reaching rebuild of their organizations; (2) yielding control over pools of resources each manager controls in order to form agile release trains is likely to face strong resistance of political nature.[4] The implementation of frameworks is not indifferent to team autonomy.[72] Conboy and Carroll warn that should an organizational transformation be based on formal frameworks, boards may be tempted by the counterproductive pursuit of compliance with a particular framework rather than the benefits brought by individual component practices.[18] On top of that, deliverables of highly regulated industries are subjected to various regulation schemes beyond typical national-level requirements, thereby

leading to a disruption of the agile process referred to as ScrumBut or even the unfeasibility of adopting a given framework.[19] This may be particularly true for entities operating within the European Union's single market, which introduces an additional layer of standardization and regulatory power.

2.2 Systematic Approaches for Agile Method Tailoring

The challenge of selecting agile practices to be adopted based on the project's characteristics is a known problem, yet there is limited guidance in the literature.[8, 58, 75] Studies on agile method tailoring usually describe the target tailored method and its success, but do not discuss the tailoring process itself in any significant detail,[35, 60, 75] whereas a few exceptions highlighted below were developed in the late 2000s with traditional agile methods in mind rather than scaling agile frameworks.

Qumer & Henderson-Sellers proposed a framework to assist managers in assessing the degree of agility they require and how to identify appropriate practices to introduce this agility into their organization.[37] The framework can be used to create, modify or tailor situation-specific agile software development processes. It makes use of a knowledge base that has been collected from different sources such as the existing agile frameworks, industrial agile adoption case studies, and agile process assessment. Esfahani et al. followed with a framework for evaluating a set of candidate agile practices to be a part of an organization-specific development method[38] The evaluation was based on the team's objectives and situation. The framework utilizes knowledge about how each practice can contribute to various project objectives, and what requisite conditions must be met for the practice to be applicable. The knowledge was collected through a systematic literature review and, likewise in the previous work by Qumer & Henderson-Sellers,[37] is represented using the i* goal-oriented modeling framework. Krasteva et al. delivered an experience-based approach to situational engineering of agile methods. Their approach supports the orderly adoption of agile practices to the software development method of an organization by reviewing research on the applicability of agile practices when particular situation factors are present.[39] It is based on an iterative and incremental process that is adaptable to project characteristics and customizable through the execution of the project. Moreover, the approach is based on four meta-models that enable the automated generation of the appropriate software development process.

All aforementioned approaches require pre-existing knowledge regarding which practices work best in which situations, yet the decision on practice selection is rarely explicit in the literature.[35] Indeed, most software development-related knowledge is tacit and resides in the brains of developers.[40] Thus, the provision of method fragment repositories imposes excessive overhead on software development organizations. Consequently, issues such as the lack of adequate information for some practices, or the mismatch of organization situation with the information that is available in the repository are inevitable. Besides, reliance on the extensive body of empirical evidence can be subject to misinterpretation, inadequate or unreliable evidential data, and conflicting scenarios (e.g., same situation, different results).[38] On top of that, each organization faces a combination of different factors operating in a unique environment, and thus practices that prove successful in one organization may not work in another in a similar context.[47, 69] Accordingly, our approach is different in the way that it simply follows the "apply-inspect-adapt" model of action, which underpins the Agile philosophy, rather than focusing on building an operational repository of method fragments and considering situational dependencies.

Even if one has an approach for method configuration, the selection of the most appropriate base method itself constitutes a challenge. The results of the research conducted by Conboy and Carroll with global companies that underwent a large-scale agile

transformation indicate that practitioners lacked an assessment model to conduct a comparative analysis among large-scale agile frameworks that could guide them in the conscious choice.[18] Fortunately, ever since they carried out their study, two such models have been established. To compare twelve scaling agile frameworks and grasp similarities between them, Diebold and his team put forward a set of criteria divided into four categories and extracted a list of practices underlying each framework.[17] In a follow-up study by Almeida and Espinheira, fifteen assessment criteria were proposed based on which they performed a comprehensive comparative analysis of six frameworks (i.e., SAFe, Scrum at Scale, Spotify, LeSS, DAD, and Nexus).[49] Both of the comparison models are complementary to our approach and might be used to prescreen existing frameworks when selecting the base method.

3 Research Design

3.1 Setting

The research was conducted at Intel Technology Poland – a Polish branch of Intel Corporation, based in Gdansk. While Intel is, first and foremost, recognized for developing microprocessors, this multinational corporation also puts great emphasis on delivering artificial intelligence and cloud computing solutions nowadays. It was one of the projects being executed by Intel, i.e., Trusted Analytics Platform, that served as the research setting in this study. This open-source project was designed to make it easier for developers to collaborate with data scientists in a cloud environment to build and operate domain-specific applications driven by advanced data analysis at scale. It integrated well-known open-source frameworks (e.g., Kafka, Spark, HDFS, Hive, and other Cloudera Hadoop components) with each other and enabled binding many microservices into a single custom workflow.

Before our research started, the project involved about sixty developers plus a management team. It was operationally handled by a Scrum Master who had made initial attempts to scale Scrum by intuitively adopting method fragments from Nexus.[26] The motivation that guided the Scrum Master in opting for Nexus was that it does not impact the organization as a whole, but confines itself to practices directly applicable to the project teams. The Scrum Master in consultation with the management team had the freedom to choose which agile methods, practices, and tools to use as long as they aligned with organizational governance structures. They were also allowed to test and experiment with the ways of working, provided they kept up with delivering value.

Within the initial implementation of Nexus, four software development teams, a team of testers, and a team of DevOps were formed. All teams used the same, single Product Backlog; however, each team maintained its individual Sprint Backlog. Each team followed Daily Scrum, Sprint Planning, and Sprint Retrospective as part of the Scrum routine. Appropriate representatives from each Scrum Team also met at the Nexus Sprint Retrospective to address shared challenges identified at the team level retros. As for Sprint Review, it was replaced by Nexus Sprint Review, where all teams met with the Product Owner and stakeholders to review the entire Integrated Increment. Besides, as an alternative to Nexus Daily Scrum, a Scrum of Scrums meeting was held once a week. Representatives from all teams attended this meeting to identify and handle integration issues.

3.2 Problem Definition

Although Intel had adopted Scrum at the team level long ago,[54] an enterprise-wide agile culture was not established at the time when we conducted the research. As a rule-bound organization, it relied on hierarchical structures and centralized control mechanisms that inherently conflicted with agility. Middle managers were stuck with traditional governance practices in terms of reporting, resource allocation, budgetary controls, and decision-making.

Lower-level managers had autonomy at the operational level, but they had to align with overarching goals set by the middle management from the US headquarters and face up to upfront project scoping, top-down changed targets as well as shifted priorities. To top it all, due to the cultural heritage of the Tayloristic past, middle and top managers were reluctant to embrace an agile mindset and dissolve their pyramidal hierarchy to facilitate shared decision-making and self-management.

In point of fact, it has been widely discussed in the literature that a large-scale agile transformation is challenging for mature bureaucratic corporations where well-established structures and routines hinder the full adoption of an "off-the-shelf" framework.[48, 49, 50, 60, 63, 64, 73] Therefore, instead of taking on an entire framework, large organizations often develop hybrid approaches depending on contextual criteria, where a customized agile method and traditional project management practices co-exist in different configurations alongside each other.[8, 50, 51, 52, 59, 64, 69, 73] However, guidance on applying any of the "off-the-shelf" frameworks outside of its prescribed process hardly exists,[18] while practitioners rarely have expertise in situational method engineering[13, 33, 35] and thus have to resort to unstructured approaches. Unfortunately, taking account of difficulties in anticipating what method fragments to choose and how to combine them effectively,[67] ad hoc tailoring usually does not bring expected software process improvements.[35] Indeed, the undertaken initiatives to tailor Nexus in Intel Technology Poland revealed the necessity of putting a more disciplined approach to work. Accordingly, a representative of the company enlisted assistance from academics, which was the genesis of the herein-reported joint venture.

3.3 Approach

As we were going to work in close collaboration with a group of practitioners, acting as a facilitator to ameliorate the immediate problem situation in their organization, naturally we chose an Action Research (AR) approach.[74] Indeed, according to Baskerville, studying the effects of specific alterations in systems development methods is the ideal domain of the AR method.[20]

During pre-study discussions with the host organization, we explained the idea of AR and its potential to support parallel academic and practical objectives. We also emphasized that as researchers, we would have the dual objective of improving scaling agile practices and at the same time, contributing to knowledge development within the participatory process. In turn, representatives of the problem owner stressed the following constraints:

- the organizational structure and governance cannot be changed;
- the steady progress in delivering the project cannot be disrupted.

Besides, we were requested to continue the scaling agile process started by the Scrum Master instead of starting from scratch.

After consulting the literature,[26, 49] we confirmed that adopting Nexus was the right decision as this framework:

- affects exclusively the operational processes;
- easily accommodates changes;
- promotes continuous improvement;
- presents low technical complexity.

On the other hand, SAFe and DAD are complex, cover the entire organization, and thus necessitate changes at all organizational levels.[10, 29, 49] Moreover, SAFe affects overall decision-making and focuses more on following predefined processes than on adapting processes to the need of the organization.[46] Furthermore, both SAFe and LeSS advocate that teams should work closely with customers,[49], whereas Trusted Analytics Platform was off-the-shelf software and hence there was no customer to work against. Finally, Spotify and Scrum at Scale are not as handy and flexible as Nexus.[49]

Since there were no conflicting expectations between both parties, we swiftly established an agreement that governed the researcher and host's involvement, collaboration, and responsibilities. As the researchers, we were responsible for guiding the overall process. Based on our prior experience with AR projects,[21] we decided to involve developers in the decision-making process ranging from adequately diagnosing the problematic situations, through suggesting and shaping solutions, to approving practices that were going to be permanently adopted in the project. Besides, we undertook to consult all planned interventions in terms of their possible conflict with the organizational governance structure before implementing them. In the spirit of Agile, our partnership was not formalized in a written contract but was based on a gentlemen's agreement.

3.4 Objectives

The project was driven by two objectives. The practical one was to assist the practitioners in informed decision-making while devising an in-house Nexus-based method. To achieve this, we helped them make gradual improvements by adding, removing, or modifying agile practices based on their appraised value and consistency with the organizational governance structure. Simultaneously, the research objective was to demonstrate how to instantiate a method configuration process using a light-way experimental approach embedded within AR cycles.

3.5 Method

As the Scrum Master had unconsciously followed the method configuration strategies by Ågerfalk et al., we decided to keep them in place. These strategies are as follows:[22]

1. selecting prescribed fragments from the base method;
2. integrating fragments from other methods to fill gaps in the base method;
3. developing new method fragments in cases when such fragments prove not applicable.

However, to provide structure for our method configuration process, we embed it in the AR framework (Figure 1). Furthermore, to avoid an overwhelming burden on staff, we employ a lightweight experimental approach based upon the "Probe-Sense-Respond" model of action defined by the Cynefin framework.[77]

Cynefin is a conceptual framework that helps decision-makers to make sense of problems and situations, in different dynamic organizational contexts, and take appropriate action.[77] It distinguishes between five types of problem contexts or "domains" (obvious, complicated, complex, chaotic, and a center of confusion) based on degrees of predictable order.[77] Each domain is provided with an appropriate model of action.[76] The problem of method configuration can be situated in a complicated domain. In this domain, there may be multiple right interventions, while the cause and effect are separated by time and space but the relationship can be revealed through investigation and expert knowledge. The model of actions best suited to this context is "sense-analyze-respond", i.e. sense incoming data, analyze that data and then respond in accordance with expert advice and interpretation of that analysis.[76, 77] As a matter of fact, all existing approaches for agile method tailoring[37, 38, 39] unconsciously exploit this model. However, building a method repository requires too much up-front investment and does not guarantee the matching between the specific situation and context at hand and the information that is available in the repository. Besides, gaining an understanding before the adoption of a new practice is inherently complex, because often the impact of one practice depends on the interaction with other practices and hence is visible after its actual enactment.[38, 55] Therefore, in carrying out the method configuration, we use multiple safe-to-fail interventions to identify the right practices. Such an approach is referred to as "Probe-Sense-Respond" and is mainly intended for the complex domain, in which the

relationship between cause and effect can only be understood in retrospect, but not in advance.[76, 77]

[Figure 1 near here]

Ultimately the proposed approach is structured according to the five phases of AR as follows:

- **Diagnosing**. The AR cycle starts with the exploration of problematic situations and the identification of their underlying causes. As for the first cycle, the researchers also have to build an understanding of the organizational and project context including organizational constraints, culture, and expectations, while both researchers and practitioner representatives must roughly familiarize themselves with existing scaling agile frameworks and practices.
- **Action Planning**. As the participating teams are expected to maintain productivity, only a manageable subset of the identified problems are considered in each cycle. We recommend starting by inspecting all deviations from the Scrum framework, which are often caused by negligence.[35] Eliminating such ScrumButs may resolve many issues[36] even without putting any scaling practices in place. The practitioners should be actively engaged in the quest for information, ideas, and practical knowledge to elaborate solutions. As a result, a set of interventions is proposed. A single intervention is one of the following types: adding a new method fragment borrowed from an off-the-shelf framework, developing a new method fragment, modifying an existing method fragment, or removing an existing method fragment.
- **Action Taking**. The devised interventions are implemented and the outcomes are observed.
- **Evaluating**. Feedback and opinions of the participating teams are collected to determine whether the intended effects of the interventions were realized and whether these effects relieved the problems. If any intervention was not successful, it can be reconsidered in the next cycle.
- **Specifying Learning**. The researchers and practitioners jointly reflect on the outcome of each intervention. Successful solutions are permanently integrated into the base method and immediately anchored in the normal way of working, while others may provide foundations for a subsequent cycle. Lessons learned are documented with the aim of informing future research and practice.

3.6 Data Sources

Both cycles were fueled by an unstructured interview with the Scrum Master, who was the person responsible for implementing and coordinating Agile practices – and thus had direct knowledge of the maturity of project management mechanisms and possible inconveniences associated with them. The first author acted as the interviewer, with the field notes being digitally drawn up on the fly. Subsequently, the initial diagnosis was completed and particularized by approaching the phenomenon under analysis from two polarized perspectives:

- *Development team perspective*, which allowed the list of challenges to be supplemented with issues of operational nature. Focus group interviews were conducted with individual teams. Members of each of the six teams were asked to identify issues in three areas: people, processes, and agile practices.
- *Management perspective*, where another focus group was set up to identify/validate root causes behind individual issues and prioritize those. Five managers were involved. Each had control over one or two development teams. On behalf of the

Product Owner, they ensured that the tasks included in the individual Sprint Backlogs were delivered.

Once interventions were conceptualized and made in the research setting, both cycles featured meticulous evaluation. All Trusted Analytics Platform developers employed at the time were approached with the questionnaire survey. Google Forms service was used. Questionnaires featured, respectively, 22 and 27 closed questions (divided into 6 and 9 themes). Those were based on five-point individual Likert-type items designed to capture to what extent respondents agreed with the statements given. On top of that, most themes provided an opportunity for expressing ideas and reservations narratively. Ultimately, 37 valid responses were put to analysis as a part of the opening AR cycle, and 28 as a part of the closing one.

4 Findings

4.1 Opening AR Cycle

The Scrum Master directed the research team's attention toward three major areas of concern, i.e., a wide range of logistical problems, indecisiveness, and poor availability of the Product Owner. The latter was also due to the fact that this individual also assumed the role of Release Manager and was responsible for liaising with part of the management team working in the USA. As for the teams, criticism that the project did not have precise customer requirements constituted one of the most frequently reported shortcomings (lack of a commercial customer meant that the requirements were derived from market observations and attempts to employ technologies that were most attractive at the time). On the other hand, the lack of detailed top-down solution architecture caused many problems with communication and integration of components delivered by different teams.

Reflection on the issues identified through *Diagnosing* and the most likely root causes behind their occurrence led to *Action Planning*. As a part of the latter, several enhancements aimed at improving the Trusted Analytics Platform venture were proposed, and then implemented and validated. Not all issues were approached within the opening AR cycle, as shown in Table 1.

[Table 1 near here]

Action Taking that spanned across a couple of three-week Sprints enabled interested parties to become familiar with all the proposed organizational changes. To collect sufficient feedback (see Fig. 2), *Evaluating* featured a multi-faceted questionnaire survey that focused on enhancements regarding Daily Scrum, Scrum of Scrums, Sprint Planning, Sprint Review, Sprint Retrospective, and Share Team Member approach, respectively. Thus, engineers working on the Trusted Analytics Platform project had the opportunity to comment on whether:

D1: Daily Meetings take place efficiently (the whole meeting is expected to take no more than 15 minutes;
D2: an appropriate number of professionals attend a Daily Meeting (i.e., 3-9);
D3: Daily Meetings need to be better organized;
SoS1: an appropriate number of professionals attend Scrum of Scrums meetings (i.e., 3-9);
SoS2: Scrum of Scrums meetings facilitate identifying issues between teams and enable their members to work together towards finding solutions;
SoS3: the frequency of Scrum of Scrums meetings is just right;
SoS4: competent people attend Scrum of Scrums meetings (i.e., those who have a full overview of the situation in their team);

SP1: only team leaders should attend the meeting where the Product Owner presents a given Sprint's priorities for each team;
SP2: the tasks that one's team performs often depend on the tasks of other teams;
SP3: it often proves difficult to plan a task a member of another team has the most comprehensive knowledge about, and the internal knowledge turns out to be insufficient to plan this task;
SP4: a developer very often takes into account the work of other teams when planning his/her tasks;
SR1: attending the Sprint Review meeting is considered useful;
SR2: the length of the Sprint Review meeting is appropriate to showcase the work done by all teams;
SR3: oftentimes, one team is unable to demonstrate the functionality they have implemented because of issues on another team;
SR4: added value of Sprint Review meetings could be bolstered by making changes to the meeting format;
R1: one often has reservations about the work of other teams;
R2: the problems flagged in the retrospectives relate to issues that affect communication between teams;
R3: a shared leadership retrospective facilitates improving team collaboration;
R4: added value of Sprint Retrospective meetings could be bolstered by making changes to the meeting format;
STM1: the Share Team Member approach has a negative impact on working in a large team;
STM2: introducing the Share Team Member approach disrupts the operation of entire teams;
STM3: having Share Team Member in place disrupts the work of individual developers.

[Figure 2 near here]

4.2 Closing AR Cycle

Since not all the problems uncovered within the opening next cycle were successfully resolved, they came back with the closing cycle's *Diagnosing*. Regardless of the above, the level of organization of Daily Scrums was poorly received. The reason for this situation was the deficient moderation of such meetings. Challenges with knowledge transfer and meetings were in fact the common denominator for many discussions at this stage. Teams quite agreed that Sprint Planning without a full picture of the situation is tricky at best. The Scrum teams and the management team agreed that the identified problems in the project set-up and inter-team communication were so far only hinted at, but no one really tried to tackle them. There was also a prevailing narrative that it was reasonable to strive for self-sufficiency of the teams, i.e., the ability for them to complete 100% of all the tasks belonging to a single problem domain. Thus, a detailed list of issues and planned corrective interventions was compiled (Table 2).

[Table 2 near here]

Again, a couple of complete Sprints were devoted to field-testing the enhancements introduced as a part of the second AR cycle. Respondents were approached with a list of questions focused on assessing the merits of the proposed solutions (see Fig. 3) that fall under Project Management, Daily Scrum, Scrum of Scrums, Sprint Planning, Backlog Grooming, Sprint Review, Sprint Retrospective, Team Rooms, and Feature Teams, respectively.

Thus, as per *Evaluating*, the research team gained knowledge on whether:

PM1: introducing the role of Release Manager facilitated the prioritization of tasks assigned to individual teams;
PM2: reducing the load of a Product Owner made him significantly more available to development teams;
PM3: reducing the load of a Product Owner led to increasing the system requirements' level of detail;
D1: separating the Scrum Master role from the Technical Lead role shortened Daily Meetings;
D2: separating the Scrum Master role from the Technical Lead role enabled the company to get more specific information from individual team members by making sure their status statements were detailed enough;
D3: status statements by team members are geared toward answering the questions what did I do yesterday? what will I do today? what is impeding my work?;
SoS1: decreasing the frequency of SoS meetings from four to two did not reduce the number of problems solved;
SoS2: decreasing the frequency of SoS meetings did not cause noticeable problems regarding the inability to flag team-wide issues;
SoS3: decreasing the frequency of meetings improved workflows within development teams;
SoS4: the meeting time limit of 30 minutes turned out to be sufficient;
SP1: planning a given Sprint during a single day proved feasible;
SP2: imposing a one-day limit on Sprint planning allowed for the faster concretization of tasks being scheduled;
SP3: imposing a one-day limit on Sprint planning bolstered interest among the participants of a meeting;
BG1: the Backlog Grooming meeting provided valuable assistance before Sprint planning;
BG2: the meeting gave team members a better perspective of what the team is expected to accomplish in the nearest future;
BG3: the meeting provided an opportunity to raise the priority of long-defined low-priority tasks;
SR1: presenting Sprint statistics at the beginning of each meeting gave a clear picture of the work performed by the teams;
SR2: confronting each team's Sprint goals with what they actually accomplished increased motivation to deliver all tasks;
SR3: fixed, predetermined order of each team's presentation increased the flow of meetings;
SR4: the intra-team Sprint Review meeting and its counterpart for the US management team were held on the same day – and it made it easier for the development teams to present their achievements;
R1: regular internal retros helped to identify issues faced by the team;
R2: strict sprint retrospective routine contributed to eliminating some of the problems identified;
TR1: placing all team members in one location improved communication within a team;
TR2: placing most project teams in one location enhanced project communication;
FT1: discontinuing Share Team Members for testers enabled better prioritization of their tasks by eliminating the need for multiple teams to complete tasks simultaneously;
FT2: upon discontinuing Share Team Members, testers were able to identify themselves with a specific team;
FT3: development teams could rely more on testers to meet sprint goals following the decision to abandon the Share Team Members approach.

[Figure 3 near here]

5 Discussion

The *Specifying Learning* phases of both cycles allowed for reflection on the relevance and effectiveness of the interventions carried out. By splitting up overly numerous teams to increase the effectiveness of communication and enhancing the routines of Daily Scrums, the scale was put in line with the benchmark from Dikert et al.[8] It also built upon the recommendations brought together by Khalid et al.[61] Whereas the validity of such a solution was confirmed, the feedback on the way this type of meeting was organized was mixed. Responses to open-ended questions positioned the lack of strong moderation as the main cause of problems in some teams, as not all meeting participants adhered to the principles presented in the Scrum Guide. The ensuing decision to restructure leadership roles made prior to the second cycle resulted in a slight improvement. Placing more emphasis on exercising self-control over the manner of one's verbal statements helped to make them more concise. This, in turn, led to maintaining better concentration during a meeting and, consequently, making it shorter. The closing AR cycle also allowed for reflection on re-structuring the roles at the managerial level. The introduction of the Release Manager role proved to be a successful change, as more clarity was achieved regarding the deadlines for functionality delivery. This intervention bridged a gap between Nexus and frameworks such as SAFe, where the responsibilities of a Product Owner do not cover release management.[16] Each team was given clear messages stating when a particular system functionality is to become part of a release. On the other hand, the expected effect regarding the better definition of backlog items due to increased availability of the Product Owner for development teams was not achieved in full.

Delegating two knowledgeable professionals from each team to Scrum of Scrums meetings allowed for better identification of issues between teams and more proficient problem-solving. The question of the number of participants attending this meeting was a hotly debated topic and became the leading target of focus groups in the closing AR cycle. Representatives of the development teams questioned the need to send two people from each team to every Scrum of Scrums meeting, being convinced that one delegate was quite sufficient. It was emphasized that with the formation of two new teams, the number of participants increased by four more people. A different standpoint was taken by the architect and the leader of the testing team, who pointed out that since more developers come to the Scrum of Scrums, the problems are solved more swiftly. It was also highlighted that the problem flagged on a given day is not likely to be solved until the next day, so meeting on a daily basis is a waste of time. The decision to reduce the number of meetings to two per week taken in the wake of these discussions did not impede reporting of problems and remained neutral to the effectiveness of problem-solving. Again, imposing a more restrictive time regime worked well in practice.

Sprint Planning was problematic from the very outset given the characteristics of the project, its complexity, and its size. As it became apparent after the opening AR cycle that teams were not thinking about tasks in the global context of the entire project, attempts were made to further improve the synchronization of work for all teams by introducing a rule to close planning within the span of one day. The benefit of doing so was that it required the introduction of Backlog Grooming during the Sprint, which worked exceptionally well. Thanks to Backlog Grooming the teams were able to better prepare for Sprint Planning and establish a common product vision more easily with an eye to what would happen in the project in the future. On the other hand, the downside of tightening the planning rigor turned out to be the fatigue of many developers with highly concentrated planning activities. Since intra-team practices inevitably emanate to other teams, it becomes valid to question the self-organization of teams in such a setting.[5] Furthermore, knowledge of a particular portion of the

project was originally scattered across several teams. This caused all sorts of disruptions to other teams by having teams communicate with each other too often.[45] The company at some stage matured to introduce Feature Teams. It is important to note the strongly differing perceptions of the interaction with the Product Owner during the planning stages. Non-leadership team members were almost unanimous in stating that having leaders present at meetings with the Product Owner was sufficient. On the other hand, people who were not technical leaders of Scrum teams, but also held important roles in the project (i.e., architect, legal officer) had a completely different opinion on this matter.

During the course of the opening research cycle, there were tasks in the project aimed at quite a significant change regarding the system architecture. Adopting a critical flashback on the large-scale agile transformation that understates technical aspects,[7] the researchers red-flagged the issue. Indeed, the change caused numerous challenges in maintaining the integration of the components that make up the platform. The developers' frustration led to blaming their failures on the other teams – the ones that were working on the architectural rework. *Specifying Learning* showed that it made sense to introduce a section in Sprint Review agendas to confront teams' plans with their achievements throughout a given Sprint. This was intended to increase the motivation of teams towards improving the way they present tasks and work to close tasks more successfully. This change, tested over the second cycle, was received with a positive response. Instituting a fixed, predetermined order in which to present the results of one's work also proved to be an impactful intervention, as it increased the efficiency with which the meeting was conducted.

Communication issues also manifested themselves at the Retrospective stage. Agile advocates against exerting top-down control,[10] hence the issue of coordination becomes more nuanced. As the project exhibited strong cross-team dependencies, the project-wide Retrospective proved crucial in driving improvements and preventing similar problems in the future. Lessons learned from the first cycle made the management team realize that they should pay more attention to this kind of meeting. Whereas positioning continuous improvement as an integral factor in enhancing coordination mechanism[2] is easier said than done, complaints that acknowledged problems were not being addressed hardly materialized in the next cycle. Another major success from a communication standpoint was succeeding in bringing both individual members and entire project teams together in a single location.

By contrast, the Share Team Member approach devised in the first cycle proved to be a dead end. It turned out that this approach presented a major inconvenience to the person who holds the "shared" team member role. Abandoning this approach in favor of Feature Teams was a welcome move. The latter is not a commonly used practice,[17] and our research provides evidence that it could provide added value in similar settings. Share Team Member-related perturbations provide supporting evidence that it is not always possible to overcome organizational resistance and introduce the promoted solution straight away because of its objective value,[6] but sometimes a company must come to certain conclusions by learning from its own mistakes. To make testers an example, it was noted that this intervention helped increase their awareness within the teams they were placed in. This, in turn, enabled better management of priorities within teams and accelerated the work of contributors. In conclusion, the closing AR cycle had a higher success rate. *Specifying Learning* came to a close with the recognition that the work comfort was steadily increasing. Thus, it was decided not to initiate another cycle, but to ensure that the measures put to work were sustained on an ongoing basis.

6 Implications of the Study

In terms of practical implications, our study equips practitioners with a lightweight systematic approach for method configuration that allows them to make up their own minds

on what is relevant in their unique situation. The proposed approach promotes the adoption of large-scale practices in an agile way with feedback loops, continuous learning, and incremental improvements. Besides, our approach is interleaved with the software development process and thus supports the continuous delivery of the product. In this way, practitioners can avoid the dramatic initial productivity slowdown reported in big bang adoptions.[47] However, we are aware that not every framework may be adopted bottom-up and tailored using the presented approach. For instance, we believe that SAFe can be successfully implemented only in a top-down, centralistic manner with commitment at all levels of an organization and a substantial upfront investment of time and resources.

Furthermore, our study makes a case for several impediments being resolved through firmly and decisively tackling ScrumButs. Companies oftentimes use convenient excuses when certain changes to implement agile principles (e.g., physically relocating project teams with limited infrastructure or politically cumbersome redesign of roles) are perfectly feasible yet require some effort. In our case, it led to many problems vanishing in a relatively short time span – even though the scaling process was only underway. Alongside fixing the Scrum process, following a structured approach should be a high priority. The staff, at least to a considerable degree, know how to improve their work practices, but require a trigger as well as facilitation to take action.

Our study advances knowledge on the process of integrating agile scaling practices with pre-existing project management practices. On the one hand, it confirms existing findings that corporate bodies tend to be rigid in nature and oppose relinquishing their authority and shifting from command-and-control management to leadership-and-collaboration.[46, 48, 50, 51] On the other hand, contrary to what is oftentimes claimed in the literature,[8, 30, 49, 46, 48] our study suggests that a successful large-scale agile transformation is possible without the adoption of an agile mindset at the organizational level and with the old bureaucracy as well as the Taylor-devised management system in place. Nonetheless, the project teams must have autonomy regarding their work practices to resolve issues associated with inter-team coordination.

Our study also adds another perspective to the theory of agile process evolution.[47, 75] It demonstrates that a gradual transition to large-scale agile at the project level (1) is possible with the preservation of traditional high-level management practices; (2) requires neither middle management involvement nor upfront investment; and (3) does not need to disrupt the continuous delivery of the product. By abolishing the dogma that "management support is a necessary condition for successful transformation",[8] our research opens new perspectives for practitioners developing multiple-team projects in rigid organizations where most managers are not ready for cultural change. Indeed, our work suggests that the essence of successful transformation is an agile mindset across development teams rather than far-reaching management support.

Our research also contributes to the body of knowledge with a more nuanced view of the context of SAFe applicability. Portraying SAFe as the most suitable framework for working in regulated, large organizations may be an oversimplification.[46, 49] As Intel is a rule-bound enterprise, we took SAFe into consideration for a while as a candidate for the base method. Although SAFe adds layers of management throughout the development process[46, 49] that should appeal to traditional managers, at the same time it requires adopting an agile culture at all levels of an organization, which embraces dissolving traditional governance practices. Therefore, as long as the top and middle managers are not open to complex adjustments across the whole organization including changes to strategy, structure, culture, and operations,[60] SAFe is not the right choice for their company.

Last but not least, our work strengthens the previous findings that large-scale agile adoption is not a one-and-done affair by just taking into use off-the-shelf, but an iterative process of stepwise refinement by context-based tailoring.[8, 21, 47, 59, 69]

References

1. Carroll N, Bjørnson FO, Dingsøyr T, Rolland KH, Conboy K. Operationalizing Agile Methods: Examining Coherence in Large-scale Agile Transformations. Lecture Notes in Business Information Processing 2020;396:75-83. doi:10.1007/978-3-030-58858-8_8.
2. Berntzen M, Moe NB, Stray V. The Product Owner in Large-Scale Agile: An Empirical Study Through the Lens of Relational Coordination Theory. Lecture Notes in Business Information Processing 2019;355:121-136. doi:10.1007/978-3-030-19034-7_8.
3. Poth A, Kottke M, Riel A. Scaling Agile on Large Enterprise Level with Self-Service Kits to Support Autonomous Teams. Annals of Computer Science and Information Systems 2020;21:731-737. doi:10.15439/2020F186.
4. Putta A, Paasivaara M, Lassenius C. How Are Agile Release Trains Formed in Practice? A Case Study in a Large Financial Corporation. Lecture Notes in Business Information Processing 2019;355:154-170. doi:10.1007/978-3-030-19034-7_10.
5. Rolland K, Fitzgerald B, Dingsøyr T, Stol KJ. Problematizing Agile in the Large: Alternative Assumptions for Large-Scale Agile Development. In 37th International Conference on Information Systems. AIS; 2016:1-21.
6. Kalenda M, Hyna P, Rossi B. Scaling Agile in Large Organizations: Practices, Challenges, and Success Factors. J. Softw.-Evol. Proc. 2018;30(10):e1954. doi:10.1002/smr.1954.
7. Pries-Heje J, Krohn MM. The Safe Way to the Agile Organization. In Proceedings of the XP2017 Scientific Workshops. ACM; 2017:18. doi:10.1145/3120459.3120478.
8. Dikert K, Paasivaara M, Lassenius C. Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review. J. Syst. Softw. 2016;119:87-108. doi:10.1016/j.jss.2016.06.013.
9. Jacobson I, Sutherland J, Kerr B, Buhnova B. Better Scrum through Essence. Softw.-Pract. Exp. 2022;52(6):1531-1540. doi:10.1002/spe.3070.
10. Edison H, Wang X, Conboy K. Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review. IEEE Trans. Softw. Eng. 2021;48(8):2709-2731. doi:10.1109/TSE.2021.3069039.
11. Rico S, Bjarnason E, Engström E, Höst M, Runeson P. A Case Study of Industry-Academia Communication in a Joint Software Engineering Research Project. J. Softw.-Evol. Proc. 2021;33(10):e2372. doi:10.1002/smr.2372.
12. Ramin F, Matthies C, Teusner R. More than Code: Contributions in Scrum Software Engineering Teams. In ICSEW'20 Proceedings. ACK; 2020:137-140. doi:10.1145/3387940.3392241.
13. Brinkkemper S. Method Engineering: Engineering of Information Systems Development Methods and Tools. Inf. Softw. Technol. 1996;38(4):275-280. doi:10.1016/0950-5849(95)01059-9.
14. Schnitter J, Mackert O. Large-scale Agile Software Development at SAP AG. Communications in Computer and Information Science 2011;230:209-220. doi:10.1007/978-3-642-23391-3_15.
15. Bick S, Spohrer K, Hoda R, Scheerer A, Heinzl A. Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. IEEE Trans. Softw. Eng 2018;44(10):932-950. doi:10.1109/TSE.2017.2730870.
16. Remta D, Buchalcevova A. Product Owner's Journey to SAFe – Role Changes in Scaled Agile Framework. Information 2021;12(3):107. doi:10.3390/info12030107.

17. Theobald S, Schmitt A, Diebold P. Comparing Scaling Agile Frameworks Based on Underlying Practices. Lecture Notes in Business Information Processing 2019;364:88-96. doi:10.1007/978-3-030-30126-2_11.
18. Conboy K, Carroll N. Implementing Large-Scale Agile Frameworks: Challenges and Recommendations. IEEE Softw. 2019;36(2):44-50. doi:10.1109/ms.2018.2884865.
19. Poth A, Jacobsen J, Riel A. Systematic Agile Development in Regulated Environments. Communications in Computer and Information Science 2020;1251:191-202. doi:10.1007/978-3-030-56441-4_14.
20. Baskerville RL. Investigating Information Systems with Action Research. Commun. Assoc. Inf. Syst. 1999;2(19):7-17. doi:10.17705/1CAIS.00219.
21. Kowalczyk M, Marcinkowski B, Przybyłek A. Scaled agile framework. Dealing with software process-related challenges of a financial group with the action research approach. J. Softw. Evol. Process 2022;34(6):e2455. doi:10.1002/smr.2455.
22. Ågerfalk PJ, Wistrand K, Karlsson F, Börjesson G, Elmberg M, Möller K. Flexible Processes and Method Configuration: Outline of a Joint Industry-Academia Research Project. In Proceedings of the 5th International Conference on Enterprise Information Systems. Escola Superior de Tecnologia do Instituto Politecnico de Setubal; 2003:519-522.
23. Malinova M, Gross S, Mendling J. A Study into the Contingencies of Process Improvement Methods. Inf. Syst. 2022;104:101880. doi:10.1016/j.is.2021.10188.
24. Karlsson F, Ågerfalk PJ. Method Configuration: Adapting to Situational Characteristics while Creating Reusable Assets. Inf. Softw. Technol. 2004;46(9):619-633. doi:10.1016/j.infsof.2003.12.004.
25. Henderson-Sellers B, Ralyté J. Situational Method Engineering: State-of-the-Art Review. J. Univers. Comput. Sci. 2010;16:424-478.
26. Schwaber K. Nexus Guide. The Definitive Guide to Scaling Scrum with Nexus. https://www.scrum.org/resources/online-nexus-guide. Published 2021. Accessed Jan 07, 2023.
27. Uludag Ö, Kleehaus M, Caprano C, Matthes F. Identifying and Structuring Challenges in Large-scale Agile Development Based on a Structured Literature Review. In IEEE 22nd International Enterprise Distributed Object Computing Conference. IEEE; 2018:191-197. doi:10.1109/EDOC.2018.00032.
28. Uludag Ö, Putta A, Paasivaara M, Matthes F. Evolution of the Agile Scaling Frameworks. Lecture Notes in Business Information Processing 2021;419:123-139.
29. Putta A, Uludag Ö, Paasivaara M, Hong SL. Benefits and Challenges of Adopting SAFe – An Empirical Survey. Lecture Notes in Business Information Processing 2021;419:172-187.
30. Laanti M, Kettunen P. SAFe Adoptions in Finland: A Survey Research. Lecture Notes in Business Information Processing 2019;364:81-87.
31. Tengstrand SN, Tomaszewski P, Borg M, Jabangwe R. Challenges of Adopting SAFe in the Banking Industry – A Study Two Years After its Introduction. Lecture Notes in Business Information Processing 2021;419:157-171.
32. Kasauli R, Knauss E, Horkoff J, Liebel G, de Oliveira Neto FG. Requirements Engineering Challenges and Practices in Large-scale Agile System Development. J. Syst. Softw. 2021;172:110851. doi:10.1016/j.jss.2020.110851.
33. Campanelli AS, Camilo RD, Parreiras FS. The Impact of Tailoring Criteria on Agile Practices Adoption: A Survey with Novice Agile Practitioners in Brazil. J. Syst. Softw 2018;137:366-379. doi:10.1016/j.jss.2017.12.012.
34. Agh H, Ramsin R. Scrum Metaprocess: A Process Line Approach for Customizing Scrum. Softw. Qual. J. 2021;29:337-379. doi:10.1007/s11219-021-09551-4.

35. Conboy K, Fitzgerald B. Method and Developer Characteristics for Effective Agile Method Tailoring: A Study of XP Expert Opinion. ACM Trans. Softw. Eng. Methodol. 2010;20(1):2. doi:10.1145/1767751.1767753.
36. Cram AW. Agile Development in Practice: Lessons from the Trenches. Inf. Syst. Manage. 2019;36(1):2-14. doi:10.1080/10580530.2018.1553645.
37. Qumer A, Henderson-Sellers B. A Framework to Support the Evaluation, Adoption and Improvement of Agile Methods in Practice. J. Syst. Softw. 2008;81(11):1899-1919. doi:10.1016/j.jss.2007.12.806.
38. Esfahani HC, Yu E, Cabot J. Situational Evaluation of Method Fragments: An Evidence-Based Goal-Oriented Approach. Lecture Notes in Computer Science 2010;6051:424-438. doi:10.1007/978-3-642-13094-6_33.
39. Krasteva I, Ilieva S, Dimov A. Experience-Based Approach for Adoption of Agile Practices in Software Development Projects. Lecture Notes in Computer Science 2010;6051:266-280. doi:10.1007/978-3-642-13094-6_22.
40. McBride T. The Use of Project Management Mechanisms in Software Development and their Relationship to Organizational Distance: An Empirical Investigation [Ph.D. Thesis]. Sydney, Australia: University of Technology; 2006.
41. Buchalcevova A, Dolezel M. Examining the Usage of Scaled Agile Methods in the Czech Republic. In ISD2021 Proceedings. AIS; 2021:1377.
42. Theobald S, Schmitt A. Dependencies of Agile Teams – An Analysis of the Scaled Agile Framework. Lecture Notes in Business Information Processing 2020;396:219-226. doi: 10.1007/978-3-030-58858-8_22.
43. Abrar MF, Ali S, Majeed MF, Khan S, Khan M, Ullah H, Khan MA, Baseer S, Asshad M. A Framework for Modeling Structural Association among De-Motivators of Scaling Agile. J. Softw.-Evol. Proc. 2021;33(8):e2366. doi:10.1002/smr.2366.
44. Butt SA, Khalid A, Ercan T, Ariza-Colpas PP, Melisa AC, Piñeres-Espitia G, De-La-Hoz-Franco E, Melo MAP, Ortega RM. A Software-Based Cost Estimation Technique in Scrum Using a Developer's Expertise. Adv. Eng. Softw. 2022;171:103159. doi:10.1016/j.advengsoft.2022.103159.
45. Özkan N, Tarhan A. A Review Of Scaling Approaches To Agile Software Development Models. Software Quality Professional 2019;21(4):11-20. doi:10.18517/ijaseit.6.6.1374.
46. Ciancarini P, Kruglov A, Pedrycz W, Salikhov D, Succi G. Issues in the adoption of the scaled agile framework. In ICSE-SEIP'22 Proceedings. ACM; 2022:175-184. doi:10.1145/3510457.3513028.
47. Julian B, Noble J, Anslow C. Agile Practices in Practice: Towards a Theory of Agile Adoption and Process Evolution. Lecture Notes in Business Information Processing 2019;355:3-18. doi:10.1007/978-3-030-19034-7_1.
48. Uludag Ö, Philipp P, Putta A, Paasivaara M, Lassenius C, Matthes F. Revealing the state of the art of large-scale agile development research: A systematic mapping study. J. Syst. Softw. 2022;194:111473. doi:10.1016/j.jss.2022.111473.
49. Almeida F, Espinheira E. Large-scale agile frameworks: A comparative review. Journal of Applied Sciences, Management and Engineering Technology 2021;2(1):16-29. doi:10.31284/j.jasmet.2021.v2i1.1832.
50. Russo D. The Agile Success Model: A Mixed-methods Study of a Large-scale Agile Transformation. ACM Trans. Softw. Eng. Methodol. 2021;30(4):52. doi:10.1145/3464938
51. Naidoo R, Rikhotso S. Balancing Autonomy and Control Tensions in Large-Scale Agile. In ISD2021 Proceedings. AIS; 2021:1317.
52. Butt SA, Piñeres-Espitia G, Ariza-Colpas P, Tariq MI. Project Management Issues While Using Agile Methodology. Lecture Notes in Business Information Processing 2022;438:201-214. doi:10.1007/978-3-030-94238-0_12.

53. Neumann M. Towards a Taxonomy of Agile Methods: The Tree of Agile Elements. In CONISOFT 2021 Proceedings. IEEE; 2021:79-87. doi:10.1109/CONISOFT52520.2021.00022.
54. Mich D, Ng YY. Retrospective games in Intel Technology Poland. In FedCSIS 2020 Proceedings. IEEE; 2020:705-708. doi:10.15439/2020F62.
55. Esfahani HC, Yu E. A repository of agile method fragments. Lecture Notes in Computer Science 2010;6195:163-174. doi:10.1007/978-3-642-14347-2_15.
56. Schön EM, Radtke D, Jordan C. Improving Risk Management in a Scaled Agile Environment. Lecture Notes in Business Information Processing 2020;383:132-141. doi:10.1007/978-3-030-49392-9_9.
57. Butt SA, Ercan T, Binsawad M, Ariza-Colpas PP, Diaz-Martinez J, Piñeres-Espitia G, De-La-Hoz-Franco E, Melo MAP, Ortega RM, De-La-Hoz-Hernández JD. Prediction based cost estimation technique in agile development. Adv. Eng. Softw. 2023;175:103329. doi:10.1016/j.advengsoft.2022.103329.
58. Neumann M. The Integrated List of Agile Practices – A Tertiary Study. Lecture Notes in Business Information Processing 2022;438:19-37. doi:10.1007/978-3-030-94238-0_2.
59. Kiv S, Heng S, Wautelet Y, Poelmans S, Kolp M. Using an ontology for systematic practice adoption in agile methods: Expert system and practitioners-based validation. Expert Syst. Appl. 2022;195(1):116520. doi:10.1016/j.eswa.2022.116520.
60. Strode DE, Sharp H, Barroca L, Gregory P, Taylor K. Tensions in Organizations Transforming to Agility. IEEE Trans. Eng. Manage. 2022;69(6):3572-3583. doi: 10.1109/TEM.2022.3160415.
61. Khalid A, Butt SA, Jamal T, Gochhait S. Agile Scrum Issues at Large-Scale Distributed Projects: Scrum Project Development At Large. Int. J. Softw. Innov. 2020;8(2):85-94. doi:10.4018/IJSI.2020040106.
62. Poth A, Kottke M, Riel A. The implementation of a digital service approach to fostering team autonomy, distant collaboration, and knowledge scaling in large enterprises. Hum. Syst. Manage. 2020;39(4):573-588. doi:10.3233/HSM-201049.
63. Stettina CJ, Hörz J. Agile portfolio management: An empirical perspective on the practice in use. Int. J. Project Manage. 2015;33(1):140-152. doi:10.1016/j.ijproman.2014.03.008.
64. Spiegler SV, Heinecke C, Wagner S. An empirical study on changing leadership in agile teams. Empirical Softw. Eng. 2021;26:41. doi:10.1007/s10664-021-09949-5.
65. Hanslo R, Tanner M. Machine learning models to predict agile methodology adoption. In FedCSIS 2020 Proceedings. IEEE; 2020:697-704. doi:10.15439/2020F214.
66. Hanslo R, Vahed A, Mnkandla E. Quantitative Analysis of the Scrum Framework. Lecture Notes in Business Information Processing 2020;376:82-107. doi:10.1007/978-3-030-37534-8_5.
67. Ambler SW, Lines M. The Disciplined Agile Process Decision Framework. Lecture Notes in Business Information Processing 2016;238:3-14. doi:10.1007/978-3-319-27033-3_1.
68. Saeeda H, Ahmad MO, Gustavsson T. Challenges in Large-Scale Agile Software Development Projects. In SAC'23 Proceedings. ACM; 2023. doi:10.1145/3555776.3577662.
69. Berntzen M, Hoda R, Moe NB, Stray V. A taxonomy of inter-team coordination mechanisms in large-scale agile. IEEE Trans. Software Eng. 2022. doi:10.1109/TSE.2022.3160873.
70. Jarzębowicz A, Ślesiński W. What is Troubling IT Analysts? A Survey Report from Poland on Requirements-related Problems. Advances in Intelligent Systems and Computing 2018;830:3-19. doi:10.1007/978-3-319-99617-2_1.
71. Hanslo R, Mnkandla E. Scrum adoption challenges detection model: SACDM. In FedCSIS 2018 Proceedings. IEEE; 2018:949-957. doi:10.15439/2018F270.

72. Gustavsson T, Berntzen M, Stray V. Changes to team autonomy in large-scale software development: a multiple case study of Scaled Agile Framework (SAFe) implementations. Int. J. Inf. Syst. Proj. Manag. 2022;10(1):29-46.
73. Marner K, Theobald S, Wagner S. Release Planning in a Hybrid Project Environment. Lecture Notes in Business Information Processing 2020;376:19-44. doi:10.1007/978-3-030-37534-8_2.
74. Staron M. Action research in software engineering. Springer International Publishing; 2020. doi:10.1007/978-3-030-32610-4.
75. Trippensee L, Remané G. Practices for Large-Scale Agile Transformations: A Systematic Literature Review. In AMCIS 2021 Proceedings. AIS; 2021:1340.
76. Childs S, McLeod J. Tackling the wicked problem of ERM: using the Cynefin framework as a lens. Rec. Manag. J. 2013;23(3):191-227. doi:10.1108/RMJ-07-2013-0016.
77. Kurtz CF, Snowden DJ. The new dynamics of strategy: Sense-making in a complex and complicated world. IBM Syst. 2003;42(3):462-483. doi:10.1147/sj.423.0462.

Table 1. Solutions elaborated during the opening AR cycle

Issue	**Root cause**	**Intervention**
Product Owner being overloaded	An individual acting as the Product Owner was assigned too many various responsibilities. Notably, Trusted Analytics Platform had him accountable for releases as well	No intervention was attempted within this cycle
Knowledge of one team member needed on another team	Knowledge of the components that make up the platform focused on individuals. Expertise (testers, DevOps) completely clustered in separate teams	Introducing the Share Team Member approach
A fair number of meetings do not contribute anything substantial	Lack of a decision-maker, thus many issues needed to be discussed in a broader group. Too many people involved in the discussion resulted in the inability to establish one common vision	Limiting meetings to solely those for which a clear purpose can be defined
Scrum teams that are too large	Too many responsibilities falling under a single team	Establishing a further two Scrum teams (breaking up the two largest teams)
Daily Scrums being overlong	Scrum teams that are too large	
Members of one team being spread across different areas of the building	Logistics problems. The legacy of previous projects	No intervention was attempted within this cycle
Challenges with identifying dependencies between individual teams' tasks	Lack of team awareness regarding tasks performed by other teams	Introducing Nexus Sprint Planning before Sprint Planning to: (1) communicate priorities; (2) discuss and select Product Backlog items for each team; and (3) make any dependencies transparent
Problems with making requirements precise	Product Owner being overloaded. Open-source nature of the project	No intervention was attempted within this cycle
Duplicated tasks	Lack of team awareness regarding tasks performed by other teams. Product Owner being overloaded	No intervention was attempted within this cycle
Failure to flag problems	Lack of developers' hands-on experience working on projects of similar complexity	Increasing the frequency of Scrum of

	levels	Scrums meetings
Low Scrum of Scrums performance	Lack of participation of professionals with high expertise	Seconding two members from each team to participate in the Scrum of Scrums

Table 2. Solutions elaborated during the closing AR cycle

Issue		**Root cause**	**Intervention**
Issues unresolved within the opening AR cycle	Product Owner being overloaded	An individual acting as the Product Owner was assigned too many various responsibilities. Notably, the Trusted Analytics Platform had him accountable for releases as well	Taking responsibility for preparing a release by a newly hired team member acting as Release Manager
	Members of one team being spread across different areas of the building	Logistics problems. The legacy of previous projects	Concentrating all members of a given Scrum team in Team Rooms. Physically moving most of the teams to a single location
	Problems with making requirements precise	Product Owner being overloaded. Open-source nature of the project	Passing control over descriptions of tasks distributed across teams to the Release Manager
	Duplicated tasks	Lack of team awareness regarding tasks performed by other teams. Product Owner being overloaded	Passing control of the product registry sequence to the Release Manager

Newly diagnosed issues	Lack of strong moderation during Daily Scrums	Focusing too much on technical aspects during Daily Scrums	Separating the responsibilities of the Scrum Master from the Technical Lead – assigning their duties to two different team members
	Too many Scrum of Scrums meetings with no impact on delivery	Too close timing between Scrum of Scrums meetings, preventing resolution of major issues	Reducing the number of Scrum of Scrums meetings to two per week. Introducing a new meeting format
	Low cross-team awareness during Sprint planning	Challenges with identifying interdependencies between teams	Introducing consultations with the Solution Architect and management team
	Problematic task planning	Knowledge of a particular project area being dispersed among members of different teams	Incorporating consultations between Scrum teams
	Inadequate amount of time allocated for teams to present accomplishments during Sprint Review	Numerous mix-ups during a meeting	Changing the format of the Sprint Review meeting
	Poor motivation for teams to prepare for Sprint Review presentations	Failure to confront Sprint goals with team performance	
	No way to address project-wide problems efficiently	Irregular Team Retrospective meetings across Scrum teams	Forcing regular Team Retrospectives. Changing the form in which those are conducted
	High workloads for professionals who work in Share Team Member mode	Subjective disadvantages outweigh the advantages of the Share Team Member approach	Discontinuing the Share Team Member approach. Introducing Feature Teams
	Sprint Planning takes too long, effectively stretching over several days	Involvement of individual team members in other activities, unrelated to planning a future Sprint	Imposing a one-day limit on Sprint planning. Setting up Backlog Grooming/Refinement sessions

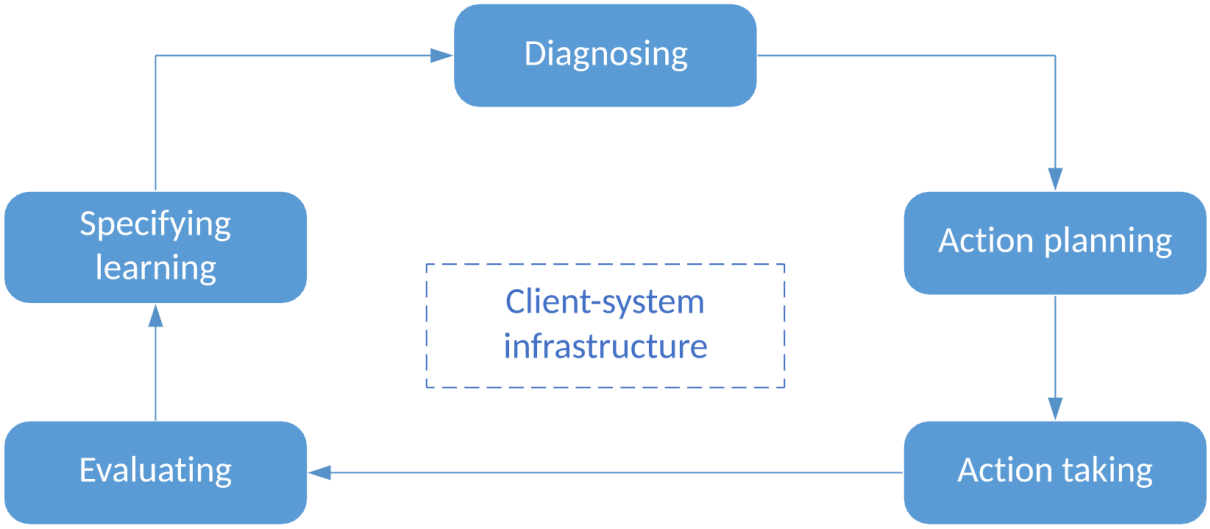

Figure 1. 5-phase Action Research design

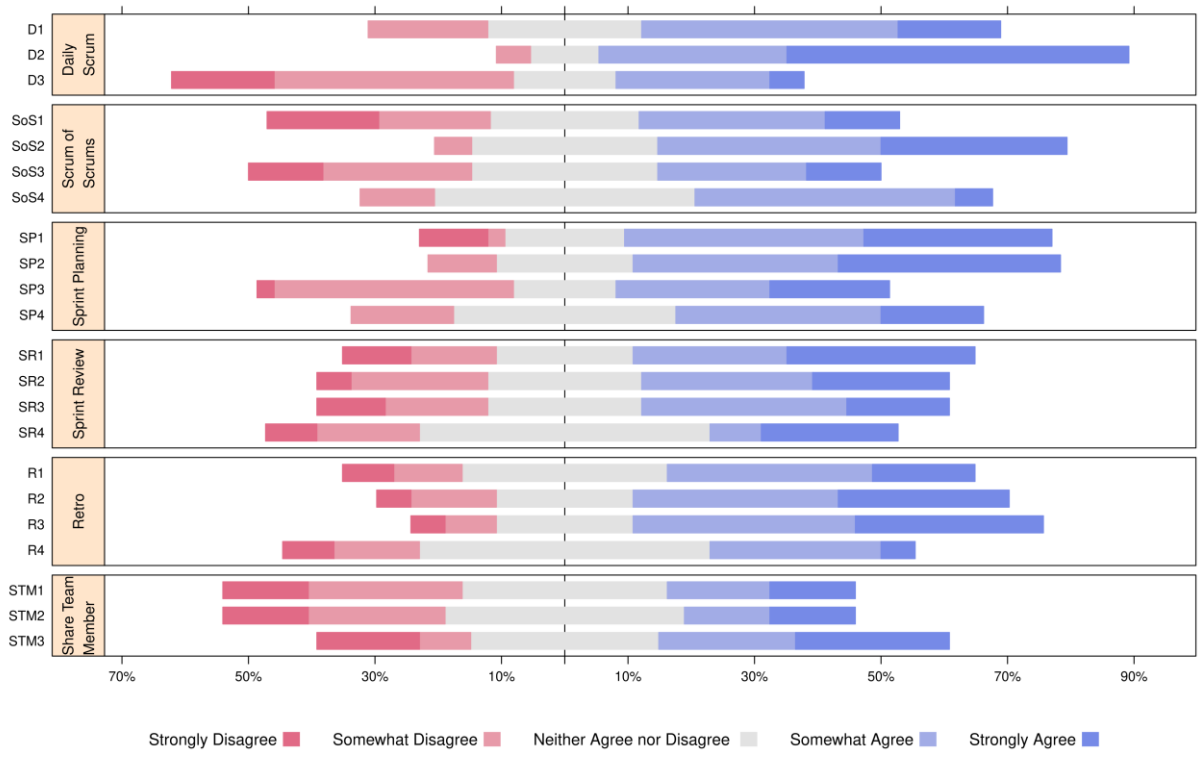

Figure 2. Evaluating the opening AR cycle

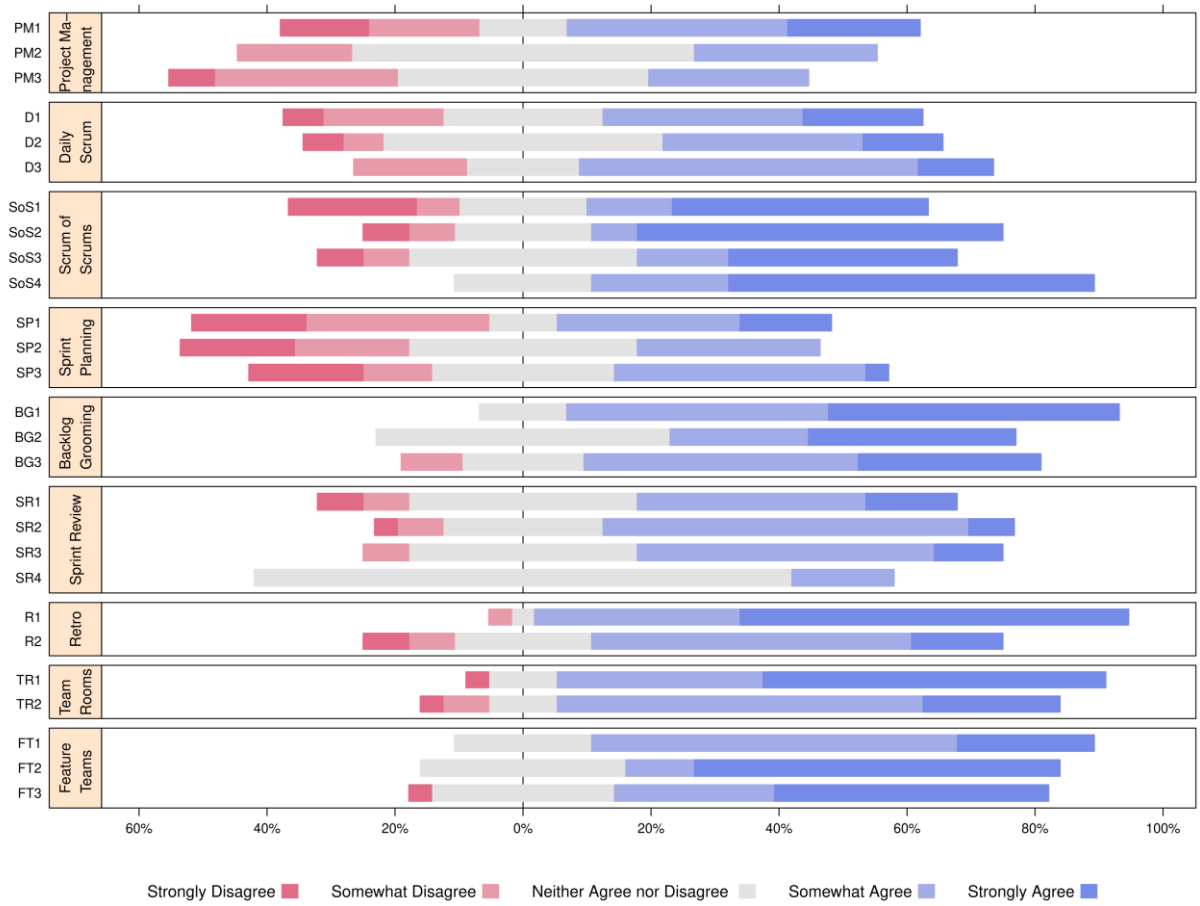

Figure 3. Evaluating the closing AR cycle