Scaled Agile Framework. Dealing with Software Process-Related Challenges of a Financial Group with the Action Research Approach[1]

Michał Kowalczyk, MSc
Nordea Bank Abp
Branch in Poland

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

Adam Przybyłek, PhD
Gdansk University of Technology
Faculty of Electronics, Telecommunications and Informatics
orcid.org/0000-0002-8231-709X

Scaled Agile Framework. Dealing with Software Process-Related Challenges of a Financial Group with the Action Research Approach

Abstract

This article reports on a domain-specific software development venture at Nordea. We explore organizational constraints, challenges, and corrective actions undertaken when scaling the agile development approach of their Core Banking Platform programme. The fit and required customizations of the rather complex and rigid SAFe framework in a policy-heavy financial institution are audited against the organic growth of the programme. In a qualitative study with three cycles using the Action Research method, both organizational and technical restrictions were tackled jointly with the company employees. Thirty-four potential solutions were developed – some of which turned out to be dead-ends. Ultimately, the effectiveness of prior recommendations was tested in a highly challenging implementation environment, and the list of guidelines was extended. We discovered several meeting and collaboration schemes that are not fully aligned with previous reports.

Keywords

Large-Scale Agile; Organizational Change; Scaled Agile Framework; IT Project Management; Action Research

1 Introduction

Ever since the release of the Agile Manifesto in 2001, agile development methods have proven to be beneficial with regards to enhancing ability to handle changing requirements, delivering software more quickly, increasing team productivity, and improving customer satisfaction.[1, 2, 58, 62, 63, 66] Although they were originally designed for small, single-team projects, success stories have inspired companies to apply them to large-scale endeavors that span over a long period[3, 15, 20, 67, 68] and within different contexts.[60] Nevertheless, the original agile methods such as Scrum, XP, or Kanban do not provide guidance on dealing with scaling issues. Thereby, their custodians as well as consultants who have helped companies in large-scale transformations, have proposed several agile scaling frameworks,[4, 57] including Large-Scale Scrum (LeSS), Scrum-at-Scale, Nexus and others – with Scaled Agile Framework (SAFe) being the most predominant.[1] These off-the-shelf solutions incorporate predefined workflow patterns to deal with issues related to large number of teams, inter-team coordination, and lack of up-front architecture.[57] One must bear in mind, though, that successful implementation of any of the aforementioned frameworks is not an easy undertaking. As a matter of fact, Abrar et al. came out with a categorization of as many as 15 demotivators for such processes from the management perspective.[7] Furthermore, numerous challenges for large-scale agile development have been identified and classified in systematic literature reviews.[4, 8, 9, 69] Hence, the call of Paasivaara and Lassenius to investigate diverse settings for scaling frameworks and to come up with practically applicable and field-tested recommendations[10] and solutions remains valid. On top of that, the bulk of reported experiences on SAFe comes from grey literature; most of the latter is published on the SAFe official website, which may lead to biased information.[15]

One of the industry sectors that is almost missing in the scientific mainstream analyzing SAFe transformations is the banking industry[3] – even though SAFe has been adopted by numerous banks (e.g., Nordea, Capital One, Standard Bank, ING Benelux, Westpac, BNP Paribas, and Deutsche Bank). We subscribe to the opinion that the scarcity of empirical studies regarding large-scale agile transformations within the banking industry is due to the fact that researchers rarely have operational access to such organizations for a longer-term transformation purpose. In turn, practitioners who have it lack tend to the

motivation to publish their unique experiences. Therefore, narrowing this gap benefits from a joint effort and a long-term collaboration of researchers and practitioners. Our empirical study follows such collaboration model and is driven by two goals: scientific and practical. The former comes down to an informed attempt to deepen the knowledge on scaling the Scrum method across large organizations and scrutinize:

RQ1: how to carry out a large-scale agile transformation based on SAFe within a regulated business environment?

RQ2: what challenges may a company encounter when adopting SAFe in such context?

RQ3: how can these challenges be addressed?

Ensuring a successful transformation of a real-world company from Scrum to SAFe constitutes the practical goal of the study. Thus, the Action Research (AR) method was used to reconcile these two goals. AR's interventions were effectuated within a large-scale IT project run by an international financial corporation – Nordea.

The main contributions of this article are threefold: (1) tangible, implementable solutions and engineering practices to tackle challenges with a SAFe transformation at a large bank; (2) empirical evaluation of prior recommendations for a large-scale agile transformation; (3) three lessons learned from our transformation:

- SAFe adoption requires knowledgeable advocates being able to articulate a compelling vision of SAFe transformation and support the organizational change;
- draw on concerns raised by employees and engage them in decision making and problem-solving processes to enhance their support for the transformation;
- consider the straightforward adoption of SAFe to be a barely first step of a successful transformation.

We believe that our experience provides valuable insights for other banks or financial institutions to establish a similar approach and tailor it to their contexts when undergoing a SAFe transformation.

Upon the introduction, a focused analysis of the related research that covers organizational change through scaling agile approaches and investigating the SAFe framework in action is conducted. The research method and setting are brought forward next. The findings of each AR cycle are followed by the recapitulation of insights and lessons learned throughout the entire venture and discussed against related research. Finally, conclusions and limitations of the study are presented.

2 Related Research

SAFe is widely considered a rather complex framework, and it incorporates a vast set of templates and process elements.[11, 6] Theobald et al. perform an inspection of practices across a wide range of relevant frameworks, which suggests that the core of widespread practices is directly derived from the generic Scrum, and place SAFe among the frameworks that feature more extensive coverage of individual practices.[12] SAFe covers the team, the entire program, and portfolio levels as well as an optional value stream level. At the team level, it adopts Scrum practices, though using Kanban is also viable. At the program level, it embarks on incremental deliveries with different scales, such as the concept of an Agile Release Train. This corresponds to sprints at the team level, but with a longer timeframe. As a process framework, SAFe also determines specific roles. Those include, for example, system team, product manager, system architect, release management team, or deployment team roles. At the portfolio level, planning is often based on Epics that define large development initiatives. The value stream level supports the development of large, complex solutions, which require multiple, synchronized releases.

While Laanti and Kettunen highlight that SAFe is renowned for its transparency and tends to replace older methods and practices, they also clearly show that in practitioners'

opinion it requires intensive custom-fitting, enlisting help in making the staff more proficient in using it, and most prominently – changing organizational mindset.[13] Dingsøyr et al. acknowledge the advice embedded in available frameworks, yet highlight the need for adapting the latter.[14] They come up with a set of 10 lessons learned that address such areas as backlog management, solution description, team coordination, testing, and continuous improvement. Putta et al. argue that whereas the challenges related to the use of the SAFe framework are generally concordant across peer-reviewed and gray sources, the same relationship is not valid when exemplifying business benefits.[15] These, in turn, are much more eagerly highlighted in the former, and undoubtedly require a deeper and unbiased insight. In another of their work, the authors focus on the process of setting up Agile Release Trains and conclude that the overall transformation is likely to be a protracted endeavor, undermined by political obstacles, and overhead with coordination activities.[16] Determinants for the use of the SAFe framework in smaller organizations were studied by Razzak et al., who concluded that SAFe practices are most effectively implemented at the team level, but at all levels, quantitative measures are inflated.[17] On the other hand, Dingsøyr et al. provide insight for agile practices usage in ventures of a very large scale.[18] They exemplify how traditional and agile approaches were mixed in such a setting and list challenges that were not successfully resolved yet – including interoperability of teams, cooperation with customer representatives as well as knowledge management.

Indeed, communication and collaboration between teams and their individual members seem to be the leading topics when discussing issues with agile upon reaching a certain scale. Having extra stakeholders in place upon scaling the Scrum approach with SAFe potentially implies changing the focus, length, and scope of meetings. That, in turn, might lead to overburdening business teams and service streams; particulars of collaboration between agile teams remain an issue as well.[19] Kasauli et al. place communication- and knowledge management-related challenges in the very center of scaled agile approaches with respect to requirements engineering.[20] Their work also brings up the issues that companies have when meeting security-attributable and other non-functional requirements. These are highly relevant given the setting of our study, since as many as approx. 50 regulation requirements in finance were revealed to impact the software development process directly.[21] Ozkan and Tarhan analyze the challenges that emerge in seven different frameworks, indicating, inter alia, that the obligatory imperative of self-organizing teams is in clear contradiction to the standardization of practices and may thus contribute to the high complexity of communication and issues with horizontal cooperation.[22, 59] The role that a Product Owner takes up regarding vertical collaboration when Scrum tends to be dependent on one's style;[23, 70] it is, however, subject to continuous improvement – and non-scheduled communication through standardized digital channels is reported to be a clear added value. Complementarily, Stray et al. focus on dependencies both between and within teams.[24] They provide a rationale for extending typical timeframes of Scrum of Scrums meetings, and for including artifacts that stimulate ad-hoc conversations (such as open workspaces featuring boards) and tools that facilitate synchronization (e.g., Skype or Slack). Bjørnson et al. set three enablers of coordination between teams against the realities of a large-scale programme, providing a rationale that while Agile practices facilitate closed-loop communication even in large-scale settings, it is harder to build trust there.[25] On top of that, the complexity of SAFe requires more effort in order for it to act as a shared mental model. Šmite et al. followed the path towards closing the research gap regarding team collaboration by scrutinizing the operation of four Spotify communities of practice.[26] They come up with a list of several challenges that are common to all the communities covered by their study along with success criteria that include (1) establishing a clear purpose, decision-making authority, and organizational sponsorship; (2) ensuring the conditions for keeping the community highly

engaged; (3) maintaining face-to-face sessions and other channels that enable regular interaction; as well as (4) driving organizational benefits from expertise and skill flow.

But the challenges with SAFe and competing frameworks cannot be simply boiled down to collaboration. Moe and Mikalsen investigate a large-scale agile transformation that accompanied building IT capabilities to enable booking ship surveys electronically in a multinational company.[27] They indicate several challenges that every transforming company should consider – highlighting the need for flexible adaptation of business models, experimenting with new products, the primacy of organizational challenges over technical ones, or the key importance of expertise for decision-making. Ericsson's experiences in requirements management across 30 cross-functional agile teams were summarized by Knauss et al.[28] They point out that requirements management environments struggle to meet two prerequisites – effective synchronization with git environments for version control and processing individual IT artifacts by multiple developers at the same time. It is the T-Reqs tool that is claimed to address key challenges in this regard. That said, we fully support the point of view that overcoming reported barriers shall not make any organization agile unless it does not come with a permanent change in the mindset of its staff and underlying culture.[29] Conboy and Carroll highlight that a particular risk of endlessly chasing near-100% compliance when adopting a given framework emerges, whereas evaluating the added value of individual practices remains a clear priority from a business perspective.[30]

Finally, Nilsson Tengstrand et al. identified some challenges that accompany a SAFe transformation in the banking industry. Many of the reported challenges overlap with the generic SAFe challenges, but two of them are specific to this highly-regulated domain, i.e., the risk of jeopardizing the trust of external stakeholders and unpredictable changes in external rules and regulations.[3]

Given all the aforementioned challenges and barriers, different research teams came up with strategic recommendations. Large enterprises are encouraged to consider implementing coaching packages and centralized agile governance while transitioning in order to overpower understanding-related barriers, manage agile-related knowledge effectively, and ensure alignment with the state-of-the-art.[31] Paasivaara and Lassenius share their insight into a successful transformation towards agile at scale: (1) decision-makers need to be prepared for discontinuing forcing their authority and be open for consent decision-making and supporting team members in taking responsibility; (2) clear guidelines ought to be in place regarding which decision falls under the community and which can be made internally by a team; (3) team capability should be enhanced by integrating experts into teams rather than enlisting their external assistance; (4) decision-making process is recommended to be shortened and adaptable based on fast feedback; and (5) the participation and input of developers who are less keen to attend community meetings should be actively sought.[32] Poth, Kottke and Riel make a case for implementing a systematic agile teamwork quality evaluation approach when employing a bottom-up approach to scaling agile practices.[33] They attempt to fill an important gap between the theoretical perception of quality and the corporate way of doing things by mapping specific indicators across widespread agile approaches to individual maturity levels. Last but not least, practitioners are advised to bear in mind that the actual transformation may exceed the timeframe of a project. Bjørnson and Dingsøyr observed an advancement of agile at scale from the first towards the second generation after a 6-month-long period.[34] A visible shift in coordination from group to individual and from scheduled to unscheduled was reported by the authors.

3 Research Approach

3.1 Method

As our study follows an organizational change, the AR approach that is commonly associated with facilitating creative solutions when managing change initiatives in real-word business settings[71] was applied. The AR constitutes a great fit when solving practical problems of companies systematically and in a cyclic manner[35] across a wide range of industries. Whereas the direct collaboration between the representatives of a company and the research team is not unanimously acknowledged within AR,[36] the vast majority of such studies involve joint interventions within real-world organizations. AR studies might be both problem-driven and research-driven. Problem-driven ventures involve enlisting academia's support to confront an unremitting business problem and inform theory in the process, whereas research-driven ones assume the existence of a theory in the first place and seeking settings that might develop and verify the theory[37] while providing practical added value. Our research falls into the former category.

AR-based studies are among the most challenging qualitative research projects to carry out successfully, as those necessitate upholding a fragile balance between real-world interventions and generating knowledge, as well as controlling highly situational research settings.[38] DeVries highlights the significance of enhancing the rigor of AR studies and introducing viable control mechanism by confronting a number of previous research initiatives with six AR-specific principles: (1) principle of foundation; (2) principle of researcher-client agreement; (3) principle of cyclical process model; (4); principle of theory; (5); principle of change through action; and (6) principle of learning through reflection.[39] While imposing additional formalisms and architectures onto the AR study is not a sufficient condition for guaranteeing its rigor, it is in most conditions the necessary one.

To enhance the scientific rigor of our research, we built upon a conceptual framework for an AR cycle by Susman[40] by superimposing new elements upon it (Figure 1). In particular, we introduce explicit knowledge transfers between research and practice as well as legitimization of the change. The legitimization is gained from employee consultation and the democratization of the evaluation of the implemented solutions. The framework consists of five interdependent phases:

- *Diagnosing*. The researchers initiate the AR cycle by building an understanding of the application domain as well as the organizational context, reality, and culture. This is an essential prerequisite for identifying primary problems triggering the need for improvements. While the organization may have already become aware of current practical problems, the researchers are in charge of conducting an independent diagnosis to confirm the nature of the problems and determine their root causes. Afterward, both researchers and practitioners jointly formulate working hypotheses regarding the research phenomenon to fuel the subsequent phases of the AR cycle.
- *Action Planning*. The researchers prepare an intervention that should address the identified problems. They reach a consensus around the intervention via employee consultation. Thereby, the practitioners are actively engaged in the quest for information and ideas to alleviate the problematic situation. Working upon the solutions is guided by the knowledge base that is assembled jointly. The practitioners bring situated, practical knowledge and experience to a table, whilst the researchers put findings from prior related studies to work and identify relevant practices from a portfolio of competitive frameworks – such as Scrum-of-Scrums, Enterprise Scrum, Spotify, LeSS, Nexus, Lean Management in this very case.
- *Action Taking*. The planned solutions are implemented driving organizational change, and, in principle, leading to an improved situation.

- *Evaluating*. Feedback and opinions of the employees affected by the change are collected. The data is analyzed jointly by the researchers and practitioners. This involves a critical analysis of the results in light of the expectations and of the practical effects that have been achieved. When the change proves successful, the evaluation must critically question whether the implemented solution was the sole cause of success.
- *Specifying Learning*. The researchers and practitioners collaborate in: (1) reflecting on the effects of the implemented solutions and outcomes of the transformation so far; (2) identifying opportunities for improvement, which serve as the starting point for a new cycle of inquiry; and (3) documenting the learning outcomes that add to the content of the knowledge base to inform both future research and practice. Successful solutions are to be permanently adopted by the organization, while solutions that have not been positively evaluated are either discarded or undergo further diagnosis across subsequent research cycle(s).

[Figure 1 near here]

3.2 Setting

Nordea Bank Abp is a Nordic full-service universal bank that manages total assets of EUR 570.35 billion (as of end of 2021) and records annual total operating income that exceeds EUR 9.6 billion.[41] Aside from Denmark, Finland, Norway, and Sweden, it operates in 12 European and 4 extra-European countries. Given the scale of the organization and historical reasons, the IT department concurrently runs several different ventures that support the business goals of the company. A vast number of product lines are maintained. The need to provide integration services to legacy applications made it difficult to understand decentralized business processes and contributed to the high overall cost of maintaining such an IT ecosystem.

Therefore, it was decided to launch a Core Banking Platform (CBP) programme targeted at implementing a brand new nCore platform deployed on UNIX clusters. All customer data from legacy mainframe systems were to be migrated. As a result, a single integrated, modern system would be in operation instead of four large systems that relied heavily on integration services. The management longed for simplifying and centralizing business processes, reducing the number of product lines considerably, bringing down the costs of maintaining services, as well as eliminating the need to employ specialists in mainframe applications, who are less and less available every year.

Whereas the implementation of Scrum principles accompanied the project from the very beginning, the team was aware of the need to introduce some effective form of coordination between teams. CBP has experimented with the use of Scrum of Scrums and explored Scrum scaling frameworks that have proven successful in large organizations. Since, in the team's opinion, the former did not guarantee achieving adequate progress in the implementation of the 5-year program (already in its 2nd year) with respect to the Management Board's expectations, the SAFe framework was selected. As part of SAFe's implementation, three Agile Release Trains were set up, each aimed at delivering a program increment. Generally, the supervision over a given train (and its individual cars) falls under the Release Train Engineer, and over the entire solution delivery – the Solution Train Engineer. Each train consists of several cars responsible for various business spheres. A given car delivers services in such a way as to be able to increment the entire IT solution – and can introduce compartments to its structure. Each car features a similar project structure, which does not deviate much from the structure of the unscaled generic Scrum. There is a single Scrum Master, Product Owner, Lead Developer, and Lead Tester per the entire car.

The velocity of the train is additionally supported by external Centers of Excellence that bring together people with similar competencies in order to spread good delivery practices.

It is Origination & Servicing that might be considered the key amongst the CBP's Agile Release Trains. It is tasked with building and maintaining banking products. The product portfolio might be very wide, as those can be designed for both individual and corporate customers across different segments. Hence, the management decided to build a train with 5 cars, each consisting of 2 compartments. As a result, a total of 10 teams that employ nearly 100 professionals are involved in delivering an increment of the target solution. Within a few months of transformational time, both advantages and issues with the SAFe framework were observed. Thus, in accordance with the Inspect and Adapt principle, a joint practitioner/researcher Action Research initiative was launched to scrutinize the issues and come up with interventions aimed at tuning in the approach and informing the theory (Table 1).

[Table 1 near here]

There is consensus among academics and professionals that to successfully implement change initiatives, there is a need for support from employees, as organizations only change through their members. Accordingly, researchers have paid much attention to the conditions under which employees are supportive of change:[42] numerous studies argue that change efforts fail when the central role that individuals play in the change process is underestimated.[43] In particular, when the change initiative comes from the management, it is crucial to explain its usefulness well or else it becomes a mandate that employees are not receptive to.[44] Employees' involvement in the change process can be achieved through participation.[45, 42] In fact, organizations are expected to give employees control over their activities and emphasize their influence on the decision-making as well as benefits for change.[46] A case can be made that it is difficult for employees to resist change initiatives in which they have been involved since its inception.

Such active involvement is often challenging to be achieved upon reaching a particular scale by an organization and, in consequence, establishing robust deployment environments, method portfolios, and communication patterns. Building on our previous experience in increasing software engineers' support for the change process we involved the employees in the transformation – ranging from actively engaging them in identifying the problematic situations, through taking their voices into account when proposing solutions, to having them evaluate the implemented solutions and co-decide which of them were worth to keep on.

Data analysis was heavily reliant on notes since initial attempts to digitally record and transcribe focus group meetings were deemed inefficient and unimplementable in the case of observations. The research team decided against printing questionnaire forms, opting for using the online Google Forms service instead. This facilitated visualizing the results of the first component of each Evaluating phase and feeding focus groups with pre-processed data more easily.

4 Findings

4.1 The 1st AR Cycle

As the most noticeable bottlenecks observed were related to the management of user requirements, the testing process, and ineffective competency matching, these issues were given the highest priority. Thus, working hypotheses WH1 through WH3 were posed. Since many diagnosed problems resulted from the division of competencies and practices not addressed in the transformation period, the implementation of the SAFe framework at Nordea

in its generic form deserves explicating. At CBP, business analysts who specialize in a specific business area are responsible for collecting detailed requirements from this area. It is the Product Owner who supervises the process of collecting requirements by said analysts and is tasked with settling the most important issues with business partners. Upon establishing a general outline of the project, the strategic goals of the company (i.e., Epics) are defined by Epic Owners. Such Epics are quintessentially backed up with a description and a business case that justifies them. Epics are established at the CBP level, which – according to many Scrum Masters at Nordea – is not a correct practice, as Epics should be defined at the entire organization level.

Epics are subsequently decomposed into Capabilities and Enablers. Capabilities constitute pieces of the functional scope that define the high-level behavior of a Large Solution and contribute a specific business value. These solution pieces are sizeable enough to require spreading their development over several Agile Release Trains and it still takes several Program Increments to complete them. Unlike Capabilities, Enablers are pieces of a Large Solution that do not deliver specific functionality/business value yet are essential to implementing Capabilities. Architectural, infrastructural artifacts, tools, compliance-related requirements, or various Proof of Concept types are covered by this category. At CBP, Capabilities and Enablers are created and prioritized by Solution Management with the assistance of Architects, Product Managers, and sometimes Product Owners. Capabilities and Enablers are then decomposed further into Features and Enablers Features within the programme area. Those differ from each other in the same way as the aforementioned higher-level artifacts. Unlike Capabilities and Enablers, however, features are delivered by a single Agile Release Train, and one increment should suffice for their implementation. Features should be created and prioritized by Product Managers, which only to some extent worked at CBP.

Given that (Enabler) Features are specified already, splitting them into Stories and Enabler Stories is performed at the team level. Product Owners should basically decompose, but it was a common practice to leave it to Business Analysts – and in the case of Enabler Features, even to developers. At CBP, the story definition process featured two stages. Tasks to be performed – being related to collecting business requirements – are scheduled and prioritized first. Product Owners and Business Analysts meet generally once a week, and the tasks are assigned to the latter. At this stage, it is often necessary to prepare high-level Use Cases for the Program Increment planning. Second, as a Use Case is about to be included in a Sprint, it is detailed down. Three Amigos Meeting is centric to the process and may involve modifying a Story. A Story is considered ready should it meet all the Definition of Ready criteria. Collecting business requirements and presenting them to developers is done in accordance with diverse practices of Business Analysts and it depends mainly on the specificity of the requirement. However, shortcomings arose in both the systematization and appropriate management of diagram documentation, and the requirements analysis process.

Insufficient maturity of individual processes in place prompted the team to define a number of problem definitions and assign them S1-S6 groups of solutions (Table 2). However, this did not peter out the Diagnosing (and thus Action Planning) phases of the first AR cycle. Nordea is a corporation, and every such large company enforces a strict security policy. Limited access to testing environments is among the consequences. The deployment of delivered IT solutions – including feeding with adequate configuration, preparing application containers, granting permissions, and the installation itself – is limited to the DevOps group. Such constraint often causes delays. Another crucial SAFe-integral aspect is Continuous Integration. In an ideal world, it boils down to a verification build being automatically triggered by any change within the code repository to see whether it caused compilation bugs or other issues. Building a project successfully should, in turn,

automatically implement the generated artifacts within the testing environments. At that stage, CBP lacked such automation and its staff massively reported instability of testing environments.

[Table 2 near here]

Solutions that were put forward to address diagnosed problems, after taking competency-related challenges into account, were field-tested throughout the entire Program Increment, and subjected to extensive *Evaluating* (comp. Table 1). Due to the corporate nature of the enterprise in which the study was carried out and no decision-making standing of the researchers, two proposed groups of solutions were dropped eventually.

Feedback on the proposed customizations (Figure 2) was canvassed from the directly involved professionals using the following questions:

- Q1: More frequent communication of a business analyst with developers of the solution and customer representatives made the feedback on the quality of the service being delivered faster compared to holding regular meetings;
- Q2: Creating a template for defining business requirements containing, among others, diagrams, information about the service recipient and necessary tips for developers facilitated the process of building the IT service significantly;
- Q4.1: The need to define a feature exclusively by a Product Manager meant that the manager did not lose control over the scope of the business area of his/her stream;
- Q4.2: Increasing the number of Product Managers across the entire programme improved the quality of defined features;
- Q5: Not being able to end a Story without testing it first increased the credibility of the banking services developed;
- Q6: The need to define a story exclusively by a Product Owner meant that the owner did not lose control over the scope of the business area of his team;
- Q7.1: Not being able to implement new functionality without writing unit tests first made the delivered service less error-prone;
- Q7.2: Putting the Pair Programming concept to work boosted the competencies of less experienced developers and led to implementing a more reliable IT solution (this question was also coded as Q8.1 and applied to evaluate the S8);
- Q8.2: Code review primarily in terms of functionality instead of stylistics led to detecting bugs within services before implementing the services;
- Q9: Providing a containerized environment for each development team accelerated the software testing process;
- Q10: Providing a separate code repository for introducing application configuration changes improved the process of continuous integration and took the strain off the DevOps;
- Q12: Setting up a website that features information regarding employees across the entire programme significantly improved the search for Single Point of Contact professionals;
- Q13: Establishing an internal StackOverflow platform enabled swifter problem solving within the software development process;
- Q14: Staging Hackathon, Innovation Jam, and Community of Practice ventures resulted in interesting ideas coming into being as well as their implementation within the organization.

[Figure 2 near here]

Evaluating revealed several still unresolved issues that potentially have a serious impact on the organization. Those include: (1) Daily Huddles being overlong; (2) no stable and understandable means for estimating story development durations; (3); the entire team not being involved in Program Increment planning; (4) straitened communication with employees working at different latitudes; (5) shortage of versatile employees; (6) deficient local database of banking products caused by difficulties in getting along with the provider of the electronic banking solution; and (7) reluctance of developers to conduct manual tests.

4.2 The 2nd AR Cycle

Based on the experience gained during the preceding cycle and after observing a number of fruitful enhancements at work, it was decided to once again look at the current situation of the company, revisit areas that have not been improved successfully, and actively seek for other areas in need of enhancement. On top of the organization under study, the research practices were streamlined as well. On one hand, not all SAFe-related customizations proposed as part of the first AR cycle turned out to be implementable/sustainable. And on the other, the researchers became more aware of the specificity of the constraints imposed by business stakeholders, and at the same time, the proposed solutions began to be more focused. *Diagnosing* brought three new working hypotheses – WH4 through WH6 (see Table 1). At a slightly later stage, issues with manual testing became apparent. Hence, nine specific problem areas were defined and assigned proposed ways to remedy them (Table 3).

[Table 3 near here]

A slightly shorter Program Increment was used to field-test a new set of solutions. In order to evaluate the interventions regarding the practices of the company (Figure 3), the CBP staff was approached with ten questions. Whereas 9 of those were typical Likert-scale questions designed to evaluate each of the proposals individually, an additional open-ended question was accounted for to prepare the ground for the focus group:

- Q1: Compulsory training of selected employees in containerized services significantly improved the process of deploying and testing applications within individual teams;
- Q2: Implementing a dedicated application bringing together functional teams streamlined the process of searching for Single Point of Contact professionals and solving a wide range of issues;
- Q3: Limiting an employee's speaking time during the Scrum meeting significantly reduced staff frustration and shortened the duration of meetings;
- Q4: Setting up meetings to dive into the estimation of Stories resulted in a far-reaching understanding of this process among employees;
- Q5: Inclusion of at least one representative of each employee position within a given team in planning resulted in improving the planning process and enhanced credibility in providing added value for a customer;
- Q6: Putting certain rules regarding the network connection in place improved the quality of communication between employees from different geographic areas;
- Q7: Setting up knowledge transfer-oriented meetings and the promotion of the T-Shape concept improved the product development process;
- Q8: Creating scripts that edit database contents after implementing each new functionality significantly reduced employee frustration and accelerated the process of developing services;
- Q9: Automating some of the manual tests unburdened manual testers and unleashed the programmers' capacity.

[Figure 3 near here]

2nd AR cycle's *Evaluating* confirmed the legitimacy of the overall approach and, at the same time, pointed at other problem areas: (1) delimiting the scope of work of individual teams; (2) bringing up technical issues during team meetings; (3) challenges with identifying errors and lacking feedback from a customer; (4) excessive number of professionals on a team; and (5) no standard for teams' operation. Since both practitioners and researchers acknowledged the room for organizational improvement that fell within the originally agreed scope of the study, an additional AR cycle was scheduled.

4.3 The 3rd AR Cycle

The final cycle was fueled by as many as 4 working hypotheses (WH7 through WH10). For some time, the company was aware that putting additional competencies to work by escalating teams has its negative sides as well. On top of that, whereas practitioners felt that adequate technological maturity had been achieved regarding the testing process, there were some indications that this particular topic had not been exhausted yet. The *Diagnosing* phase of the third cycle brought up an extremely serious problem – within the programme, tasks were now and then orphaned, and bottlenecks appeared in non-priority streams. In the former case, a kind of procedural debt emerged: whereas new procedures in this area proved nearing the ones the stakeholders longed for, the disruptions can be traced back to the transformational period. On the other hand, after a deeper analysis, the inter-team rivalry turned out to be the decisive factor in the latter case. Table 4 introduces problem definitions along with specific actions to deal with those problems.

[Table 4 near here]

In line with the practices used in previous AR cycles, a complete Program Increment was used for *Action Taking* and laying the foundations for *Evaluating* the interventions. 11+1 questions were asked, the feedback to which is visualized in Figure 4:

- Q1: Including competing teams in the same Agile Release Train resulted in a drastic reduction in their rivalry for the same resources;
- Q2: Setting up team representatives' meetings devoted to scrutinizing the scope resulted in the clear identification of team members with the newly assigned areas of responsibility and introduced a structured order of scope at the programme level;
- Q3: Splitting large teams into smaller ones improved the team management process and vastly reduced employee frustration;
- Q4: Implementation of the Ways of Working standard for individual teams of the same Train eliminated the problem of different ways for interpreting the same processes;
- Q5: Introducing periodic Defect Triage Meetings enabled reducing the number of recurring defects and facilitating the process of defining them;
- Q6: Introducing periodic Scrum of Scrums Meetings enabled improving the process of synchronizing the workflows of individual teams under an Agile Release Train;
- Q7: Introducing Handover Meetings enabled improving communication between the programmers and the staff tasked with the deployment of the solution and reducing the number of errors when introducing configuration changes;
- Q8: Introducing N-core Triage Meetings streamlined the process of identifying errors and their proper selection;
- Q9: Implementing Continuous Exploration noticeably improved the process of searching for new functionalities and collecting requirements;

Q10: Introducing Problem Spaces Meetings enabled streamlining the process of decomposing the system architecture and vastly improved the process of prioritizing tasks and tracking their progress;
Q11: Setting up Solution Sync Meetings at the management level significantly contributed to improving the escalation process of the revealed problems across the entire programme.

[Figure 4 near here]

Whereas the final cycle brought the Scaled Agile Framework used in the organization much closer to high maturity, some problems went beyond the AR initiative. These include, in particular, universal challenges of the IT community and those that result from restrictive corporate policies. Thus, the activities accomplished as a part of *Evaluating* highlighted: (1) the shortage of versatile employees; (2) a small share of remote work; (3) failure to pass on sufficient knowledge and documentation by an employee who leaves the company; (4) accumulation of documentation that is out of date; (5); rare team integration meetings; (6) too complicated and bureaucratic process of submitting requests for access to certain programme infrastructure resources; and (7) grounds for appointing a team responsible for quality control of meetings held in the organization.

4.4 Specifying Learning

Regarding the overall architecture of the SAFe framework, the participants of the study firmly and unanimously found reducing the scope of Agile Release Trains and introducing a new role, i.e., a Product Manager, to be highly convenient solutions. The latter is responsible for managing the backlog of Features and the flexibility of the framework makes him/her optional – it is not necessary to implement everything perfectly according to the scheme in a given organization. Nevertheless, in Nordea's case employing the right person as a Product Manager enabled better control over the scope of the project, whereas reducing the scopes that fall under CBP's Trains allowed Product Managers to work more efficiently. Such a change significantly improved the process of understanding customers' needs, helped to define the business goals of the organization correctly, and, in fact, to estimate the expenditures/workloads that will be needed to develop the product. Hence it can be considered a vital action behind the correct transformation.

All the Scrum Masters shared the opinion that Capabilities became a very helpful artifact during the implementation of the project. Capabilities allowed the priorities of the entire programme to be passed over to Agile Release Trains, making it easier to plan the division of labor and increase the reliability of delivery. Should the organization be dealing with a part of the system that is supposed to be delivered across several Program Increments, the Capabilities turn out to be the perfect "box" with instructions on what and how ought to be packed inside by employees of various departments so that it can be seamlessly delivered to a customer at a later stage. A standout feature of capabilities is that due to the high level both their business goals and hierarchy is defined, the need arose to integrate not only among cars of a given Train but also with other Trains. That instituted communication between different, distant functional teams.

As the transitional period is concerned, participants of the focus group meetings subscribed to the point of view that in Nordea's case, the key was to certify a dozen or so Scrum Masters towards SAFe – who could then train the rest of the relevant staff and pass domain-related knowledge around. It was acknowledged that such SAFe evangelists are inclined to serve the company throughout the entire transformation period, which can save the organization under transformation a lot of money that it might as well spend on expensive

contractors. This solution also guarantees that each employee shall receive the necessary support to be able to find oneself in the new work configuration. Therefore, a vital lesson learned from the study is that:

"*SAFe adoption requires knowledgeable advocates being able to articulate a compelling vision of SAFe transformation and support the organizational change*"

The feedback collected throughout the study strongly highlighted that it is vital to stick to the limited number of professionals within individual teams, as it became clear that when the number of team members gradually increases, the coordination and cooperation within the team plummets in efficiency. Dividing a team into smaller ones is a must, where it is much easier and more sustainable to control the workflows of employees from the very beginning. And, if necessary, a new team ought to be set up just-in-time.

When discussing the SAFe enhancements and organizational customizations, Scrum Masters stressed that testing and closing Stories proved to be a persistent issue during the transformation. It was mainly related to the poor performance of servers behind testing environments, which were flooded with many requests, as well as the practice of creating Enabler Features to test a Story. The approach worked out, which might be taken advantage of universally across large companies, is to set up a separate environment for each development team, thanks to which closing Stories and delivering new functionalities should be much faster. Moreover, while a Story is acknowledged as a point of reference for estimating the effort,[64] not only the number of Story Points for its development ought to be considered – but also a certain number of Story Points allocated for testing. So, for instance, if a Story was reckoned at 5 Story Points, then decomposing it further to a specific share of those points assigned to building a service/product and providing the development with an overhead of testing-related points to reach 5 is required.

Another change that is closely related to the use of Scrum in large organizations is putting the teams that compete for resources in the same Agile Release Train. In smaller organizations – just as at the infancy stages of the programme development – controlling integration between teams is not as much of a challenge, hence the experience of most professionals in this regard was limited. The escalation made load management and priority-related issues noticeable. The teams responsible for supplying semi-products to these competing teams unknowingly favored some of them, forgetting to take care of the others in their schedule. The solution discussed met with high levels of appreciation at CBP and was applied in several cases, significantly reducing the level of staff frustration.

Contrary to the researchers' presumptions, widening the spectrum of meetings at various levels caused no noticeable side effects or employee opposition. Solution Sync turned out to be of central importance with respect to CBP operations. Putting it into effect enabled synchronization between Agile Release Trains of the just-introduced nCore platform, operational timing between both solution architects and product decision-makers, as well as maintaining the solution delivery process. Solution Sync provided for analysis of the current statistics for the solution and the program and made controlling various events coming within the Solution Delivery block possible. Within this meeting, the statuses of Capabilities and Features were also updated, and it was ensured that previously escalated problems could be clarified or escalated even higher. Solution Sync is a kind of meeting that should take place in every large organization that seeks high-quality and sustainable communication. As expected, arranging Scrum of Scrums meetings led to an improvement in cooperation between dependent teams. In addition, serendipitous synergy was achieved. Common problems of teams whose Scrum Masters took part in the meeting were solved jointly and spontaneously. Owing to focusing solely on tasks that may have an impact on other teams during such meetings, it was possible to quickly identify various issues that would not be noticeable at

other types of meetings. So, it was decided to continue holding these meetings twice a week. Given the above, the study clearly guides agile scaling policymakers to:

"*draw on concerns raised by employees and engage them in decision-making and problem-solving processes to enhance their support for the transformation*"

Naturally, people tend to oppose changes – and unlike the Solution Sync, some of those enhancements had to be implemented despite initial reservations. Compulsory training of selected employees in the field of service containerization significantly improved the process of implementing and testing applications within individual teams. Each team delegated at least one professional who was tasked with preparing the team's testing environment for the latest changes once a week. Should accelerating this process be desired or simply a need to introduce rapid changes in the environment arose, the person responsible for the deployment pulled changes from the development branch where the changes were put. Thanks to the organizational enhancements made, the process of delivering added value to the client has been significantly accelerated since a Story could now be closed much faster with no need to wait for it to be successfully implemented within testing environments.

The turbulent beginnings of the transformation process and the number of organization-specific improvements and tailor-fittings of SAFe after its initial implementation make it clear that such scaling frameworks cannot be treated as out-of-the-box products. Every organizational change must be carefully considered to ultimately bring added value to the organization, prolonging the implementation process. Ultimately, lesson learned here is to:

"*consider the straightforward adoption of SAFe to be a barely first step of a successful transformation*"

One of the changes that were strongly emphasized in internal discussions was the automation of some manual tests. The workshops held for programmers by seasoned testers made the former a bit more enthusiastic about writing automatic tests. Owing to this, many manual testers have been relieved of their current duties and could support the team on other fronts, and the work of developers became slightly more diversified. This enhancement also affected the scope of work and task planning during the Sprint to a degree. From this moment on, a certain amount of time must also be taken into account for an additional activity that did not classify as a functional value of a final CBP product.

5 Discussion

The systematic literature review by Dikert et al. revealed that the research on large-scale agile transformations was lagging behind the state-of-the-practice at the time.[8] Although there have been several experience reports and case studies since then, guidelines on how to scale agile methods to large-scale projects still leave a lot to be desired. In this research article, we elaborate on and build upon the work by Paasivaara et al. who presented four recommendations for large-scale agile transformations: (1) use an agile mindset and take an experimental approach to the transformation; (2) implement the transformation stepwise; (3) divide large products into product areas, in which teams can specialize; and (4) use a common agile framework for the whole organization.[47]

Not only we followed their lead, but we also developed a new guideline: involve employees in the change process at each step to enhance their support or acceptance for change initiatives. For the Scrum scaling venture to be successful, employee involvement and the ability to actively influence a change is essential. This is a critical success factor even for organizations heavily reliant on corporate policies. Our guideline addresses one of the greatest challenges to scaling Agile, which is resistance to change.[1, 3, 8, 11, 15, 27, 47, 65] We

derived this guideline directly from organizational change theories. Effective organizational change is recognized in theory as the sum of three concepts: (1) the need for change being understood by employees; (2) the employees' acceptance for the proposed solution; and (3) the employees' participation within the change process.[48, 50, 8] Accordingly, in our AR initiative, we framed organizational changes as opportunities to advance and to renew the organization in order to make it more successful. Furthermore, we ensured employees' involvement in the change process through intense participation, which is in line with the principles of AR. Each implemented practice or solution was assessed by employees after the period of use. Only changes that had been approved by employees were permanently adopted by the organization.

This empirical study tells that SAFe – with all its rigidity being criticized, for instance, by Schwaber[51] – constitutes a good fit for highly formalized and policy-heavy financial institutions. By tackling the research questions that accompany large-scale agile transformations within regulated business environments, we were able to deliver and field-test several solutions that are universal in nature and to provide a high-level mapping of previously reported agile at scale challenges and possible mitigators (Table 5). We reckon equipping the follow-on organizations with means to capitalize on a catalog of ready-made tailor-fittings and thus to shorten the transformation process in equally formalized settings to be an important practical implication of the research. Putting employees' voices first helped us to downplay two challenge categories reported by Edison et al.[4] On the other hand, the need to break up overexpanding teams without paying much attention to short-term rationale came through strongly in our study. This confirms the results of the selective literature review by Khalid et al.[52] Whereas the necessity to introduce an additional layer of coordination[53] found adequate support in our research, Backlog Grooming is definitely not enough to avoid bottlenecks in such a setting. This was clearly demonstrated by the diagnosed competitive imbalance related to requests for delivering semi-products and the appropriation of service streams by some teams.

[Table 5 near here]

Last but not least, the feedback from the conducted study constitutes a voice in opposition to considering additional meetings in terms of a challenge by e.g., Uludağ et al.[54] Of course, we would suggest assigning a high priority to a comprehensive review of communication practices and infrastructure in place, as some of them initially felt short of expectations given the considerable responsibility and intensity of communication inherent to scaled-up Scrum. Nevertheless, we believe that one of the key principles of the original Agile Manifesto, i.e., face-to-face conversations being the most productive way of conveying information,[55] became somewhat outmoded despite being held in high esteem.[26, 61] This is mainly due to the high efficiency of the virtual communication schemes and environments in place upon their tuning, which at the same time prepared the programme to operate in COVID-19 conditions.

6 Conclusions

6.1 Summary of the Findings

The joint research effort of practitioners and researchers enabled a number of organizational enhancements, the vast majority of which have found their permanent place in the host company. As part of the three strictly interconnected action research cycles, more than thirty of the proposed changes were implemented successfully out of 34 proposed. Most of the modifications went beyond custom fitting the generic SAFe framework, but nevertheless significantly contributed to improving the quality of banking services provided

and, to some extent, assisted in the transformation from Scrum to SAFe. The bulk of the solutions had to be developed internally since SAFe just defines a pattern of operation and does not provide specific practices. Said transformation is not over yet, as it is a complex process and minor enhancements take place from time to time. Nevertheless, all the steps undertaken so far brought Nordea much closer to making the new project methodology mature. The Core Banking Platform program has been around for over four years. The phase is slowly approaching, in which the main staff activities will be related to the maintenance of existing solutions. Therefore, some of the implemented SAFe solutions may not be used as intensively as at the beginning of the transformation. On the other hand, experiences of the CBP are highly likely to be a subject of diffusion across related programmes.

It should be noted that not all working hypotheses were confirmed by the study. Notably, WH3 stating that the organization requires setting up internal support platforms for developing competencies and exchanging information on them was decisively rejected. On top of that, support for WH8 was not undisputed. Whereas major technical problems were indeed dismissed from the testing process and the stakeholders could prioritize addressing organizational inefficiencies, not all hopes related to containerization were met. Watching the S9 solution at work throughout not only the first but the succeeding cycles as well confirmed that corporate policies in place kept the contents of a database on a provided lightweight container below its potential, as there was no green light for connecting to a more abundant base from the test environment.

6.2 Limitations and Future Work

The entire personnel taking part in the study was aware of the research setting, and feedback was reliant on self-reported questionnaires. The former introduces the *interaction of testing and treatment* threat to internal validity. The latter in turn potentially exposes the findings to the risk associated with subjectivism coming from anticipations and professional backgrounds of individual participants. Therefore, threats to construct validity cannot be ruled out. Due to the highly international nature of the company and the employment of professionals with strongly diverse backgrounds, the study is less prone to cultural bias. Nevertheless, the non-negligible buildup of workflows behind the Core Banking Platform programme at the Polish site constitutes, in our judgment, a research limitation.

Due to the natural focus of the Action Research studies towards a single business entity, the study is accompanied by fairly similar threats to external validity as typical of case studies of a single-case design. One must be aware that by embedding the research process within a large financial organization, attempts to generalize research results should be made with much caution. Given strict policies in place, the way SAFe is implemented in a banking company shall differ to a degree in particular stages and the form of implementation against companies from other sectors. Therefore, the study might constitute a good basis for future cross-sector comparisons. On top of that, attempts to convey the lessons from implementing SAFe in large-scale organizations to medium-sized businesses require, in our opinion, re-visiting some artifacts of this framework. Thus, exploring the validity of targeted artifact simplifications is one of the threads of our future research.

Data Availability Statement

All the relevant data are included in the body of the article.

References

1. Digital.ai. 15th annual state of agile report. https://info.digital.ai/rs/981-LQX-968/images/RE-SA-15th-Annual-State-Of-Agile-Report.pdf. Published 2021. Accessed Mar 05, 2022.

2. Dybå T, Dingsøyr T. Empirical studies of agile software development: a systematic review. *Information and Software Technology*. 2008;50(9-10):833-859. doi:10.1016/j.infsof.2008.01.006.
3. Nilsson Tengstrand S, 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. doi:10.1007/978-3-030-78098-2_10.
4. Edison H, Wang X, Conboy K. Comparing methods for large-scale agile software development: A systematic literature review. *IEEE Transactions on Software Engineering*. 2021. doi:10.1109/TSE.2021.3069039.
5. Rupp C. *Scaling Scrum across modern enterprises: Implement Scrum and Lean-Agile techniques across complex products, portfolios, and programs in large organizations*. Birmingham: Packt Publishing Ltd; 2020.
6. Knaster R, Leffingwell D. *SAFe 4.5 distilled: Applying the Scaled Agile Framework for lean enterprises*, 2nd Edition. Boston, MA: Addison-Wesley Professional; 2018.
7. 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. *Journal of Software: Evolution and Process*. 2021;33(8):e2366. doi:10.1002/smr.2366.
8. Dikert K, Paasivaara M, Lassenius C. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*. 2016;119:87-108. doi:10.1016/j.jss.2016.06.013.
9. Uludag Ö, Kleehaus M, Caprano C, Matthes F. Identifying and structuring challenges in large-scale agile development based on a structured literature review. In *EDOC 2018 Proceedings*. IEEE; 2018:191-197. doi:10.1109/EDOC.2018.00032.
10. Paasivaara M, Lassenius C. Scaling scrum in a large globally distributed organization: A case study. In *ICGSE 2016 Proceedings*. IEEE; 2016:74-83.
11. Kalenda M, Hyna P, Rossi B. Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process*. 2018;30(10):e1954. doi:10.1002/smr.1954.
12. 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.
13. Laanti M, Kettunen P. SAFe adoptions in Finland: A survey research. *Lecture Notes in Business Information Processing*. 2019;364:81-87. doi:10.1007/978-3-030-30126-2_10.
14. Dingsøyr T, Dybå T, Gjertsen M, Jacobsen AO, Mathisen TE, Nordfjord JO, Røe K, Strand K. Key lessons from tailoring agile methods for large-scale software development. *IT Professional*. 2019;21(1):34-41. doi:10.1109/MITP.2018.2876984.
15. Putta A, Paasivaara M, Lassenius C. Benefits and challenges of adopting the scaled agile framework (SAFe): Preliminary results from a multivocal literature review. *Lecture Notes in Computer Science*. 2018;11271:334-351. doi:10.1007/978-3-030-03673-7_24.
16. 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.
17. Razzak MA, Richardson I, Noll J, Canna CN, Beecham S. Scaling agile across the global organization: An early-stage industrial SAFe self-assessment. In *ICGSE 2018 Proceedings*. IEEE; 2018:116-125.
18. Dingsøyr T, Moe NB, Fægri TE, Seim EA. Exploring software development at the very large-scale: A revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering*. 2018;23(1):490-520. doi:10.1007/s10664-017-9524-2.

19. 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.
20. Kasauli R, Liebel G, Knauss E, Gopakumar S, Kanagwa B. Requirements engineering challenges in large-scale agile system development. In *RE'17 Proceedings*. IEEE; 2017:352-361.
21. Poth A, Jacobsen J, Riel A. A systematic approach to agile development in highly regulated environments. *Lecture Notes in Business Information Processing*. 2020;396:111-119. doi:10.1007/978-3-030-58858-8_12.
22. Ozkan 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.
23. 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.
24. Stray V, Moe NB, Aasheim A. Dependency management in large-scale agile: a case study of DevOps teams. In *HICSS 2019 Proceedings*. University of Hawaii; 2019:7007-7016.
25. Bjørnson FO, Wijnmaalen J, Stettina CJ, Dingsøyr T. Inter-team coordination in large-scale agile development: A case study of three enabling mechanisms. *Lecture Notes in Business Information Processing*. 2018;314:216-231. doi:10.1007/978-3-319-91602-6_15.
26. Šmite D, Moe NB, Levinta G, Floryan M. Spotify guilds: how to succeed with knowledge sharing in large-scale agile organizations. *IEEE Software*. 2019;36(2):51-57. doi:10.1109/MS.2018.2886178.
27. Moe NB, Mikalsen M. Large-scale agile transformation: A case study of transforming business, development and operations. *Lecture Notes in Business Information Processing*. 2020;383:115-131. doi:10.1007/978-3-030-49392-9_8.
28. Knauss E, Liebel G, Horkoff J, Wohlrab R, Kasauli R, Lange F, Gildert P. T-reqs: Tool support for managing requirements in large-scale agile system development. In *RE'18 Proceedings*. IEEE; 2018:502-503.
29. Ebert C, Paasivaara M. Scaling agile. *IEEE Software*. 2017;34(6):98-103. doi:10.1109/MS.2017.4121226.
30. Conboy K, Carroll N. Implementing large-scale agile frameworks: Challenges and recommendations. *IEEE Software*. 2019;36(2):44-50. doi:10.1109/ms.2018.2884865.
31. Poth A, Kottke M, Riel A. Scaling agile on large enterprise level – Systematic bundling and application of state of the art approaches for lasting agile transitions. In *FedCSIS 2019 Proceedings*. IEEE; 2019:851-860.
32. Paasivaara M, Lassenius C. Empower your agile organization: Community-based decision making in large-scale agile development at Ericsson. *IEEE Software* 2019;36(2):64-69. doi:10.1109/MS.2018.2886827.
33. Poth A, Kottke M, Riel A. Agile team work quality in the context of agile transformations – A case study in large-scaling environments. *Communications in Computer and Information Science*. 2020;1251:232-243. doi:10.1007/978-3-030-56441-4_17.
34. Bjørnson FO, Dingsøyr T. Transitioning from a first generation to second generation large-scale agile development method: towards understanding implications for coordination. *Lecture Notes in Business Information Processing*. 2020;396:84-91. doi:10.1007/978-3-030-58858-8_9.
35. Merriam SB, Tisdell EJ. *Qualitative research: A guide to design and implementation*, 4th Edition. Hoboken, NJ: John Wiley & Sons; 2015.
36. da Cunha PR, de Figueiredo AD. Action-research and critical rationalism: A virtuous marriage. In *ECIS 2002 Proceedings*. Association for Information Systems; 2002:19-27.

37. Avison D, Baskerville R, Myers MD. The structure of power in action research projects. In Knock N, ed. *Information Systems Action Research*. Boston, MA: Springer; 2007:19-41.
38. Avison D, Baskerville R, Myers MD. Controlling Action Research projects. *Information Technology & People*. 2001;14(1):28-45. doi:10.1108/09593840110384762.
39. DeVries EJ. Rigorously relevant action research in information systems. In *ECIS 2007 Proceedings.* Association for Information Systems; 2007:n49.
40. Susman G. Action research: a sociotechnical systems perspective. Morgan G, ed. *Beyond Method: Strategies for Social Research*. Newbury Park: Sage; 1983:95-113.
41. Nordea Bank Abp. Nordea annual report 2021. https://www.nordea.com/en/doc/annual-report-nordea-bank-abp-2021.pdf. Published 2022. Accessed Mar 05, 2022.
42. Przybyłek A, Albecka M, Springer O, Kowalski W. Game-based sprint retrospectives: Multiple action research. *Empirical Software Engineering*. 2022;27:1. doi:10.1007/s10664-021-10043-z.
43. Choi M. Employees' attitudes toward organizational change: A literature review. *Human Resource Management*. 2011;50(4):479-500. doi:10.1002/hrm.20434.
44. Spayd MK. Evolving agile in the enterprise: implementing XP on a grand scale. In *ADC'03 Proceedings*. IEEE; 2003:60-70.
45. Goulet PK, Schwieger DM. Managing culture and human resources in mergers and acquisitions. In Stahl GK, Björkman I, ed. *Handbook of Research in International Human Resource Management*. Cheltien: Edward Elgar Publishing; 2006:405-429.
46. Rodríguez-Sánchez JL, Mercado-Caruso N, Viloria A. Managing human resources resistance to organizational change in the context of innovation. *Smart Innovation, Systems and Technologies*. 2020;167:330-340. doi:10.1007/978-981-15-1564-4_31.
47. Paasivaara M, Behm B, Lassenius C, Hallikainen M. Large-scale agile transformation at Ericsson: A case study. *Empirical Software Engineering*. 2018;23(5):2550-2596. doi:10.1007/s10664-017-9555-8.
48. Lenberg P, Tengberg LGW, Feldt R. An initial analysis of software engineers' attitudes towards organizational change. *Empirical Software Engineering*. 2017;22(4):2179-2205. doi:10.1007/s10664-016-9482-0.
49. Rafferty AE, Jimmieson NL, Armenakis AA. Change readiness: A multilevel review. *Journal of Management*. 2013;39(1):110-135. doi:10.1177/0149206312457417.
50. Tiong TN. Maximising human resource potential in the midst of organisational change. *Singapore Management Review*. 2005;27(2):25-36.
51. Schwaber K. UnSAFe at any speed. https://kenschwaber.wordpress.com/2013/08/06/unsafe-at-any-speed. Published 2013. Accessed Mar 05, 2022.
52. Khalid A, Butt SA, Jamal T, Gochhait S. Agile Scrum issues at large-scale distributed projects: Scrum project development at large. *International Journal of Software Innovation (IJSI)*. 2020;8(2):85-94. doi:10.4018/IJSI.2020040106.
53. Jha MM, Vilardell RMF, Narayan J. Scaling agile scrum software development: providing agility and quality to platform development by reducing time to market. In *ICGSE 2016 Proceedings.* IEEE; 2016:84-88.
54. Uludağ Ö, Kleehaus M, Dreymann N, Kabelin C, Matthes F. Investigating the adoption and application of large-scale scrum at a German automobile manufacturer. In *ICGSE 2019 Proceedings.* IEEE; 2019:22-29.
55. Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin RC, Mellor S, Schwaber K, Sutherland J, Thomas D. Principles behind the Agile Manifesto. https://agilemanifesto.org/principles.html. Published 2001. Accessed Mar 05, 2022.

56. Misra S. Pair Programming: An empirical investigation in an agile software development environment. *Lecture Notes in Business Information Processing.* 2021;408:195-199. doi:10.1007/978-3-030-67084-9_13.
57. Uludağ Ö, Putta A, Paasivaara M, Matthes F. Evolution of the agile scaling frameworks. *Lecture Notes in Business Information Processing.* 2021;419:123-139. doi:10.1007/978-3-030-78098-2_8.
58. Zamudio L, Aguilar JA, Tripp C, Misra S. A requirements engineering techniques review in agile software development methods. *Lecture Notes in Computer Science*. 2017; 10408:683-698. doi:10.1007/978-3-319-62404-4_50.
59. Ozkan N, Tarhan A. Evaluation of Scrum-based agile scaling models for causes of scalability challenges. In *ENASE 2020 Proceedings*. Scitepress; 2020:365-373. doi:10.5220/0009390403650373.
60. Marcinkowski B, Gawin B. A study on the adaptive approach to technology-driven enhancement of multi-scenario business processes. *Information Technology & People*. 2019;32(1):118-146. doi:10.1108/ITP-03-2018-0142.
61. Jarzębowicz A, Sitko N. Communication and documentation practices in agile requirements engineering: A survey in polish software industry. *Lecture Notes in Business Information Processing*. 2019;359:147-158. doi:10.1007/978-3-030-29608-7_12.
62. Hanslo R, Mnkandla E, Vahed A. Factors that contribute significantly to Scrum adoption. In *FedCSIS 2019 Proceedings*. IEEE; 2019:821-829. doi:10.15439/2019F220.
63. Hanslo R, Tanner M. Machine learning models to predict agile methodology adoption. In *FedCSIS 2020 Proc*eedings. IEEE; 2020:697-704. doi:10.15439/2020F214.
64. Butt SA, Misra S, Luis SDMJ, Emiro DHF. Efficient approaches to agile cost estimation in software industries: A project-based case study. *Communications in Computer and Information Science*. 2020;1350:645-659. doi:10.1007/978-3-030-69143-1_49.
65. 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.
66. Misra S, Omorodion FM, Damasevicius R. Metrics for measuring progress and productivity in agile software development. *Advanced Intelligent Systems*. 2021;1372:469-478. doi:10.1007/978-3-030-73603-3_44.
67. Gustavsson T, Bergkvist L. Perceived impacts of using the scaled agile framework for large-scale agile software development. In *ISD 2019 Proceedings*. AIS; 2019: article 11.
68. Abrar MF, Khan MS, Ali S, Ali U, Majeed MF, Ali A, Amin B, Rasheed N. Motivators for large-scale agile adoption from management perspective: A systematic literature review. *IEEE Access*, 2019;7:22660-22674. doi:10.1109/ACCESS.2019.2896212.
69. Marinho M, Camara R, Sampaio S. Toward unveiling how SAFe framework supports agile in global software development. *IEEE Access*. 2021;9:109671-109692. doi:10.1109/ACCESS.2021.3101963.
70. Ramin F, Matthies C, Teusner R. More than code: contributions in Scrum software engineering teams. In *ICSEW'20 Proceedings*. ACM; 2020:137-140. doi:10.1145/3387940.3392241.
71. Gapp R, Fisher R. Achieving excellence through innovative approaches to student involvement in course evaluation within the tertiary education sector. *Quality Assurance in Education*. 2006;14(2):156-166. doi:10.1108/09684880610662033.

Table 1. AR architecture employed

Cycle	Phase	Scope	Data coll. techniques	Participants
I	*Diagnosing*	**WH1**: Documentation of business and functional requirements is heterogeneous and burdened with diffused responsibility **WH2**: Corporate restrictions on access to the test environment and cursory code review abate the effectiveness of testing **WH3**: The organization requires setting up internal support platforms for developing competencies and exchanging information on them	Observation	• 2 Business Analysts • 3 Scrum Masters • Researcher
	Action Planning	Elaborating a list of 14 solution groups assigned to individual problem areas that were identified	Unstructured interview	
	Action Taking	Complete program increment that encompassed 9 two-week-long Sprints (ending with June 25, 2019) was accomplished. 12 of 14 planned solution groups were implemented	Observation	• 3 Scrum Masters • Researcher
	Evaluating	14 questions (5-degree Likert scale) that cover 12 solution groups implemented as a part of *Action Taking*. 7 identified organizational issues. Major reservations feedback regarding 3 solution groups were recorded. A follow-up focus group interview to identify the root causes behind negative feedback was held	Questionnaire survey	• 10 developers per team, 3 teams of the same Agile Release Train
			Focus group interview	• 3 members of each team (Analyst, Scrum Master, Programmer) • Researcher
	Specifying Learning	WH1 and WH2 accepted, whereas WH3 – rejected. The initial master plan to put two AR cycles into effect was confirmed	Observation	• 2 Business Analysts • 3 Scrum Masters • Researcher
II	*Diagnosing*	**WH4**: Large agile teams are likely to have difficulties with backlog item estimation as well as progress tracking **WH5**: The geographic dispersion of the organization introduces challenges in terms of communication and human resource management **WH6**: The test environment is burdened with keeping test data being fed up-to-date	Observation	• 3 Scrum Masters • Researcher
	Action Planning	Elaborating a list of 9 specific solutions targeting identified problems one-by-one	Unstructured interview	
	Action Taking	Complete program increment that encompassed 8 two-week-long Sprints (ending with November 29, 2019) was accomplished. All planned solutions were implemented	Observation	
	Evaluating	Respondents were approached with 9 questions	Questionnaire	• 10 developers per

		(5-degree Likert scale) supported by a single open-ended question. 5 organizational issues were identified. Addressing minor reservations regarding 3 solutions in course of a follow-up focus group interview	survey	team, 3 teams of the same Agile Release Train
			Focus group interview	• 7 most experienced members per team; 2 teams of the same Train • Researcher
	Specifying Learning	WH4 through WH6 accepted. The decision to extend the study for an additional AR cycle was made	Observation	• 3 Scrum Masters • Researcher
III	*Diagnosing*	**WH7**: The progress of the project and the organizational changes that follow it cause further escalation of team sizes, diverging codes of conduct, coordination issues, and resource-related rivalries **WH8**: Technical problems were dismissed from the testing process, yet it is not free from organizational inefficiencies **WH9**: Discontinued forms of documenting the requirements from the early transitional period led to some mocks finding their way into the production release **WH10**: Teams focus mainly on their own goals instead of contributing to the success of the entire program	Observation	• 3 Scrum Masters • 3 Testers • Agile Release Train engineer • Researcher
	Action Planning	Elaborating a list of 11 specific solutions targeting identified problems one-by-one	Unstructured interview	
	Action Taking	Complete program increment that encompassed 8 two-week-long Sprints (ending with March 03, 2020) was accomplished. All planned solutions were implemented	Observation	
	Evaluating	Respondents were approached with 11 questions (5-degree Likert scale) supported by a single open-ended question. 7 organizational issues were identified. Benefits and risks of proposed changes were scrutinized further in course of a follow-up focus group interview	Questionnaire survey	• 10 developers per team, 3 teams of the same Agile Release Train
			Focus group interview	• A total of 20 members of the same Train • Researcher
	Specifying Learning	WH7, WH9, and WH10 were accepted. WH8 was only partially accepted	Observation	• 3 Scrum Masters • 3 Testers • Agile Release Train engineer • Researcher
			Focus group interview	• 3 Scrum Masters • Research team

Table 2. Solutions elaborated during the 1st AR cycle

ID	Problem definition	Proposed solution(s)
S_I_1	Data analysis not performed thoroughly: • lack of systematic approach leads to overlooking vital details or to ordinary misinterpretations and, in effect, flawed information reaching final business requirements specifications; it may cause significant functional, non-functional, or even integration-level issues at the implementation stage • both parties being forced to communicate in a language that is not their mother tongue makes pinpointing complex topics closely related to the business aspects of electronic banking difficult at times • a customer may have some vision of the target IT solution, but is unable to state one's expectations regarding immediate future	• continual communication with the customer aimed at the successive collection of up-to-date requirements • a strict policy of inquiry in case of uncertainty • incorporating a list of standardized questions as an auxiliary technique • consulting any change in the business documentation with developers • setting up meetings between a business analyst, developer, and the customer should a requirement necessitate clarification
S_I_2	Business documentation not detailed enough: • behavior and some functionality not captured even in a minimalist diagram form • missing information on who is the go-between and target customer of a given service • valuable information on various banking services is sometimes not documented in the place where it should be, but in the Intranet and mailboxes instead	• delivering a template that enforces the process of creating business documentation
S_I_3	"Full SAFe" variant being implemented on CBP level: • Full SAFe is intended for entire organizations • designing an additional management level for assessing competitive advantages of individual Epics did not bring business value as the priorities followed long-term planning and Roadmap 2020 anyway • the variant in place requires delegating Business Analysts to support Epic Owners in preparing Epics	• switching the SAFe implementation in CBP from "Full SAFe" to "Large Solution SAFe"
S_I_4	Features being created by Business Analysts or Product Owners: • extensive scope and internal complexity of the project prevents a Product Manager from acquiring all the substantive knowledge about various types of products and processes of electronic banking in a reasonable timeframe • acting in lieu of a Product Manager by peers reduces the likelihood of bottlenecks but introduces risks of him/her losing control over the scope of the project and business primacies being inadequately reflected by Feature prioritization	• a policy of creating Features exclusively by Product Managers • a smaller and more homogeneous range of Agile Release Train activities
S_I_5	Creating Enabler Features for time-consuming tasks that do not involve coding: • using Enabler Features to cover test-oriented workflows imposes the risk of separating the Story coding stage from testing it, whereas this should be done in a single Sprint and should essentially feature an iterative process	• the compulsion to link the coding and testing of the same Story more closely

	• the delay between finalizing coding by a developer and the delivery of development to the test environment is significant (it can be up to 3 days), hence there is practically no chance to confine a comprehensive story to a single Sprint	
S_I_6	Defining Stories by Business Analysts: • whereas refining Stories itself can be done by team members, when Business Analysts start creating Stories on their own – often without consulting a Product Owner – the owner may sometimes lose control over one's backlog and may not be able to prioritize everything well • relying solely on analysts' declarations (who do not need to have comprehensive knowledge of the scope of the team's work) may lead to underestimating the business importance of Stories or, in turn, may overestimate them due to the lack of a complete understanding of their contents	• a policy of defining Stories exclusively by Product Owners
S_I_7	Reduced efficiency of isolating error on test servers: • after deploying application components on production environments, all too often errors that have not been detected in test environments were revealed • the frustration of both developers and testers	• implementing Test-Driven Development • writing unit tests for existing code in a consequent manner • putting the Pair Programming concept[56] to work
S_I_8	Code review primarily in terms of stylistics: • when reviewing code written in a high-level language, the functionality itself and the principles of the object-oriented paradigm were sometimes considered of secondary importance • teams struggled with consistent application of SOLID, "Do not Repeat Yourself" (DRY), or "Keep It Simple, Stupid" (KISS) principles	• code review both in terms of functionality and stylistics, with emphasis on the former
S_I_9	Instability of testing environments: • updating/repairing/reinstalling/restarting a specific instance of the test environment results in a temporary inability to test the software • postponing testing in time hinders delivering the promised functionality and the contractor' reliability suffers	• testing the software on the local environment or container issued for a team
S_I_10	Application configuration on the DevOps side: • the necessary information on the configuration of various applications is stored in unambiguous templates, but changing such configuration requires a special request to the DevOps group • DevOps awareness of the impact of a given change on the entire environment is limited, as such things are generally known to the developers of a given solution	• enabling developers to introduce configuration changes in a separate configuration environment
S_I_11	Launching a remote project build manually: • less-experienced developers tend to forget launching the build and are often not aware whether they damage code modifications of their peers • unsuccessful launch of the build only when it needs to	• launching a remote project build automatically

	be executed (one of the main conditions for the code to be merged into the main branch), due to e.g., a non-anticipated upgrade of the provider API's version or suddenly merging approximate functionality of another branch leads to time-consuming verification of logs • a successful verification build should be a trigger for a professional tasked with the pre-merge Code Review to review that code being aware that it will not interfere with other developers' code	
S_I_12	Inefficient search for Single Point of Contact professionals: • in a situation where contact with the team responsible for some part of the integration is needed, it is unlikely that any member of such team can directly assist in resolving the issue • number of query redirects builds up	• setting up a website that covers all CBP and associated external teams along with information on the competencies of their members
S_I_13	Developers struggling with finding an answer to immediate questions related to organizational IT: • developers need to efficiently adapt to changes within the project • newly hired employees when getting acquainted with the project do not know where to look for general information and answers to specific IT-related issues	• establishing an internal StackOverflow platform
S_I_14	No meetings to develop programming and architectural competencies: • developers are inherently specialized in a specific area, but they might have a blurred understanding of the architecture of the banking system or networking services created by their peers • a wider business-IT environment, which in this type of organization is highly complex, is also poorly recognized at the specialist level	• funding a cyclical Community of Practice event, where topics that address various aspects of electronic banking shall be discussed • organization of Hackaton meetings

Table 3. Solutions elaborated during the 2nd AR cycle

ID	Problem definition	Proposed solution(s)
S_II_1	A tight cycle of improvement and versioning of banking applications requires experienced staff who will constantly supervise and manage the process at the level of the team's containerized test environment: • no institutionalized knowledge flow from containerized environments experts to novices, which is focused on the process of implementing new applications and maintaining them	Designating and training employees who will be responsible for implementing changes in the team's containerized test environment
S_II_2	The internal StackOverflow platform did not cope with finding answers to bothering questions efficiently: • a significant lag between stating a problem and obtaining an answer existed since questions were directed to the entire developer-level staff, whereas individual employees involved in daily duties assumed feedback being provided by the others	Setting up groups of functional teams in the selected communicator that would enable providing feedback by an expert who is available at the time

S_II_3	Daily Scrum being overlong: • such meetings lasted up to an hour, and nothing pointed at streamlining and keeping reins on these activities • the number of professionals in teams cannot be reduced by decomposing teams due to the project specificity	Introducing means aimed at reducing the speaking time of each team member
S_II_4	There was no unified and comprehensible form of effort estimation for Stories: • it happens that an employee knows how much (more or less) time he/she can spend on completing a task, yet the concept of a Story Point seems to be something not duly defined in Scrum and thereby difficult to adapt in the daily process of planning a project • the level of abstraction that results from the Story Point estimation can be perceived and interpreted one way or another by different people, which makes such a method the bane of all team members instead of being approachable and easy to use	Finding a new form of estimating Stories or setting up practical workshops to clarify the Story Point estimation
S_II_5	The entire team not being involved in planning a Program Increment: • Scrum Masters, Software Architects, and Business Analysts were primarily involved in the process • no participation of the majority of programmers and testers questioned the sense of such activity and could potentially lead to reduced credibility and lack of timely delivery of services	Accounting for at least one representative of each employee position within a given team in the planning process
S_II_6	Hindered communication with employees working at different latitudes: • communication between people from different geographical areas requires appropriate tools thanks to which it will be possible to communicate efficiently over long distances • when staff tried to set up a teleconference with non-European employees, many a time the latency level was so high that it was impossible to communicate operationally	Organization of more frequent team colocations or investment in a network connection with superior Quality of Services parameters
S_II_7	Scarcity of highly versatile staff: • conducting an in-depth diagnosis of a specific, complex problem required the involvement of numerous field specialists at one time and place, who will inspect individual parts of the process	Setting up knowledge transfer-oriented meetings to reduce the number of professionals involved in solving given issues in the long run
S_II_8	The deficient local database of banking products that can be traced back to struggling in getting along with the provider of the electronic banking solution: • the data generated and modified across various types of tests are lost when deploying a new release, as the provider of the banking system did not update the database with prior changes along with the release and provided its original contents only	Establishing a process pipeline that would run scripts responsible for adding, modifying, or deleting records in the database
S_II_9	The reluctance of programmers to conduct manual testing: • management decided to prioritize familiarity with the solution over manual testing skills during recruitment • programmers postulated to delegating more responsibilities to testers, which sparked conflicts between those groups	Automation of what is tested manually to the largest extent possible

Table 4. Solutions elaborated during the 3rd AR cycle

ID	Problem definition	Proposed solution(s)
S_III_1	The rivalry between teams for the same resources: • finite resources within the CBP include, first and foremost, streams responsible for the infrastructure or building integration with back-end systems (e.g., ledger- or data warehouse-related) • the capacity of this stream was somewhat monopolized, therefore in some areas it was not possible to deliver fully functional solutions regardless of internal progress	Including competing teams in the same Agile Release Train and orienting service provision towards trains instead of individual teams
S_III_2	Challenges related to determining the scope of work of individual teams: • CBP's transitional period was initiated at the level of the project structure, without in-depth reflection on the substantive content that should be dealt with by the newly created teams • there were contents of the solution's scope that none of the teams/trains wanted to take care of and disputes emerged over which team should accept them within its scope • arbitrary allocation of disputed contents by management without consulting interested teams resulted in staff frustration and extensive delays in delivering individual components and the entire system	Setting up team representatives' meetings devoted to scrutinizing the scope
S_III_3	Too many professionals on individual teams: • the organic development of the project that involves directly adding newly hired employees to existing teams' iteration after iteration leads to some teams reaching a scale of up to 25 people • this is contrary to the Scrum nature as well as makes it difficult to communicate and manage the entire venture	Splitting large teams into smaller ones that might be either assigned new Scrum Masters or fall under shared coordination by existing ones
S_III_4	No standards regarding working practices of teams in place: • since each team had its own rules of operation (quite divergent and not written down explicitly) it was difficult for professionals from outside the team to familiarize themselves with the progress of a given team because the same data had to be interpreted differently • for instance, heterogenous statuses for Defects and Stories as well as different workflows for these items resulted in the *in-progress* status meaning in one case that the item was being analyzed, whereas for another team its development was already taking place, as the team stipulated an additional *in analysis* status	Creating a set of high-level principles and working rules that arrange the governance of individual teams with the general concept of Agile Release Train operation by implementing the Ways of Working concept at the team level
S_III_5	Challenges regarding identifying duplicate Defects and analyzing sets of errors: • whereas determining whether a specific case is a system error or an intended feature of the system does not pose a significant problem, verifying whether a given defect has already occurred or not is more challenging and, unfortunately, may lead to duplicating of a defect • defect backlog management is far trickier in companies that feature many components and system dependencies • just knowing that a defect already exists is not enough; simply tagging a ticket incorrectly results in the defect not	Introducing periodic Defect Triage Meetings focused on identifying areas in the system that could be streamlined and improving ticketing skills

	being displayed for staff that should have access to such information	
S_III_6	Issues with synchronizing the workflows of dependent teams: • team members are primarily interested in the backlog of their team and less in tasks and general situation that takes place in other employee groups • there was no stable control over inter-team dependencies, which manifested itself in certain services under development being delivered much too early or too late	Introducing periodic Scrum of Scrums Meetings devoted to relationships between teams that involve Scrum Masters, Product Owners, program stakeholders, and key consultants from external companies
S_III_7	Incorrect implementation of components/configuration changes: • even though a repository in which developers can deposit all kinds of configuration changes was established, there are still some areas of the organization where the deployment team (per corporate policy) must set up new functionalities in testing or even production environments on their own • no assistance from the creators of the banking solution sometimes led to the implemented changes not being verified correctly and the risk of departing from a programmer's intentions	Upon each new release, detailed information regarding deployment must be provided in dedicated templates, and Handover Meetings associating the entire team responsible for delivering a specific service along with staff tasked with deployment must be held
S_III_8	Challenges with identifying system errors: • the CBP program features tools for exploring logs as well as application servers that support a wide range of ways to collect the history of service use • despite this, at times the dedicated staff dealing with the verification of logs is forestalled by a phone call from the final customer alerting about a faulty functionality	Introducing N-core Triage Meetings, where business can report a bug and set up an incident that is subsequently delegated seamlessly to relevant teams under the Agile Release Train
S_III_9	Imprecise business requirements: • the form of collecting system requirements from the infancy stages of CBP, which in a sense resembled the prototyping process, led to the delivery of only a fragment of what was expected from the final solution • evolution of the approach resulting from previous improvements eliminated most of the weak points, but the practice of collecting requirements is still closer to an iterative rather than continuous phenomenon	Implementing Continuous Exploration that accounted for just-in-time delivery of use cases and derivatives following the roadmap agreed with the stakeholders
S_III_10	No exact flow of operation of the integrated part of the system when defining Capabilities and Features in place: • despite having knowledge and skills to imagine how a given Feature works, there is often no information on the integration and final purpose of other Features • knowledge regarding an isolated component is insufficient when confronted with the vague understanding of the hierarchy of Features and the general application of the Capability it consists of	Introducing Problem Spaces Meetings devoted to division of architecture, prioritization of works, and monitoring their progress
S_III_11	Inability to deliver a specific part of the product during the Program Increment: • CBP at the time lacked mechanisms to keep customers informed of potential delays in delivering scheduled components	Weekly Solution Sync, i.e., a meeting of all Release Train Engineers, the Solution Train Engineer, the Solution Manager, Delivery Managers, and Test Managers to address current risks and problems in achieving the goals of the ongoing Program Increment

Table 5. High-level mapping of adopting SAFe and solutions in highly regulated settings

Challenge category	**Solutions that fall into the category**
Communication & Coordination	S_III_2 S_III_4 S_III_5 S_III_6
Management & Organization	S_II_6 S_III_1
Software Architecture	S_I_9 S_I_10 S_I_11
Requirements Engineering	S_I_2 S_I_4 S_I_6 S_III_9
Customer Collaboration	S_I_1 S_II_8 S_III_8
Method Adoption-Related Challenges	S_I_3 S_I_5 S_II_5 S_III_10 S_III_11
Team-Related Challenges	S_I_6 S_II_3 S_III_3
Education & Training	S_I_14 S_II_1 S_II_4
Knowledge Management	S_I_12 S_I_13 S_II_2 S_II_7
Quality Assurance	S_I_7 S_I_8 S_II_9 S_III_7

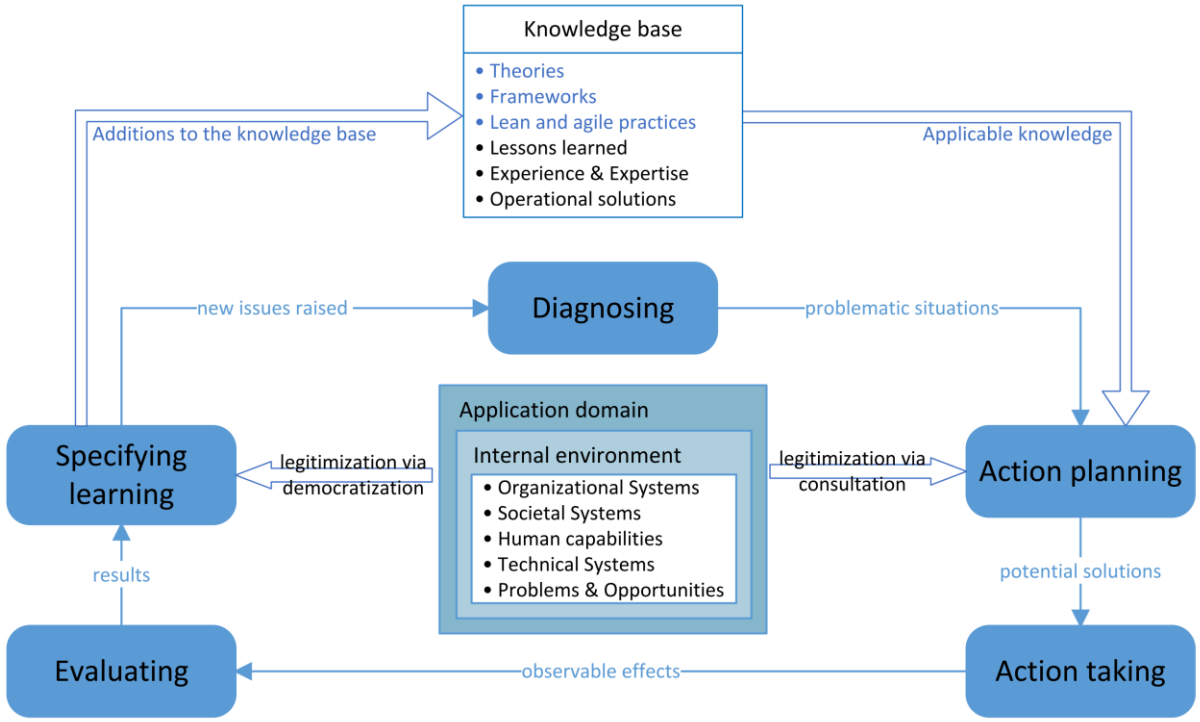

Figure 1. Research framework

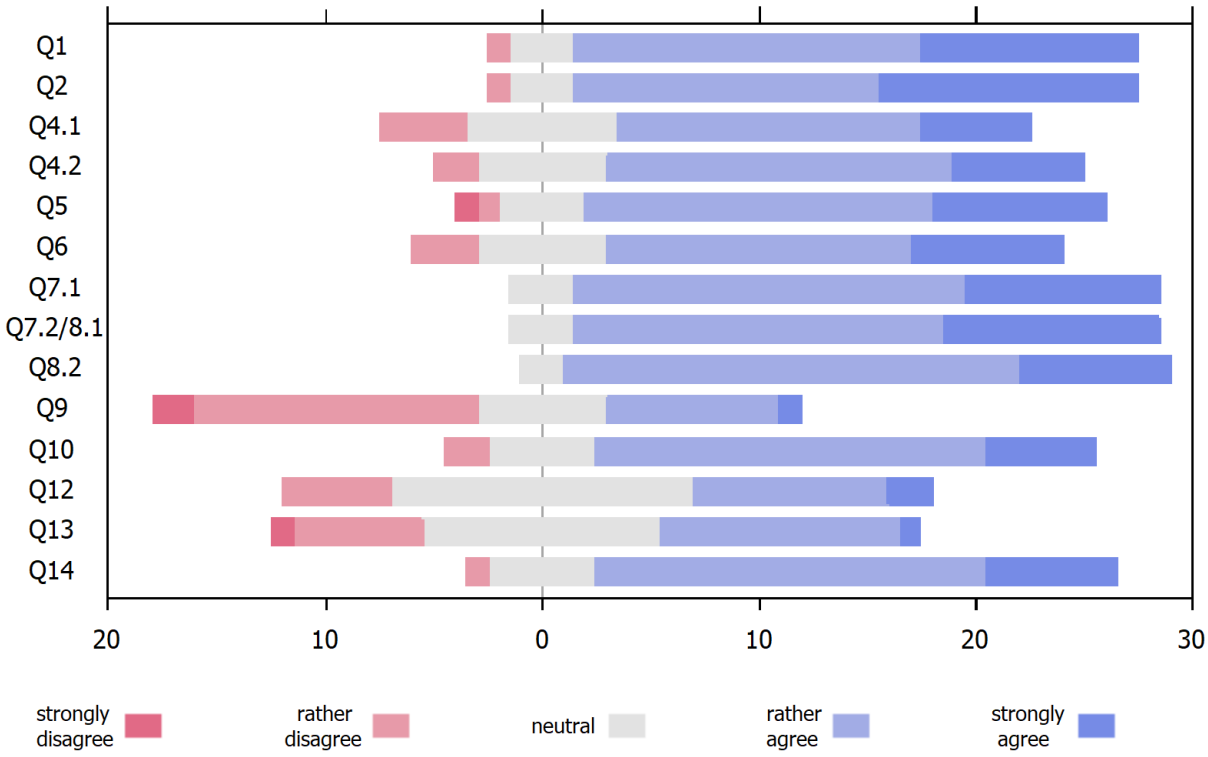

Figure 2. Feedback on the proposed customizations; the 1st AR cycle

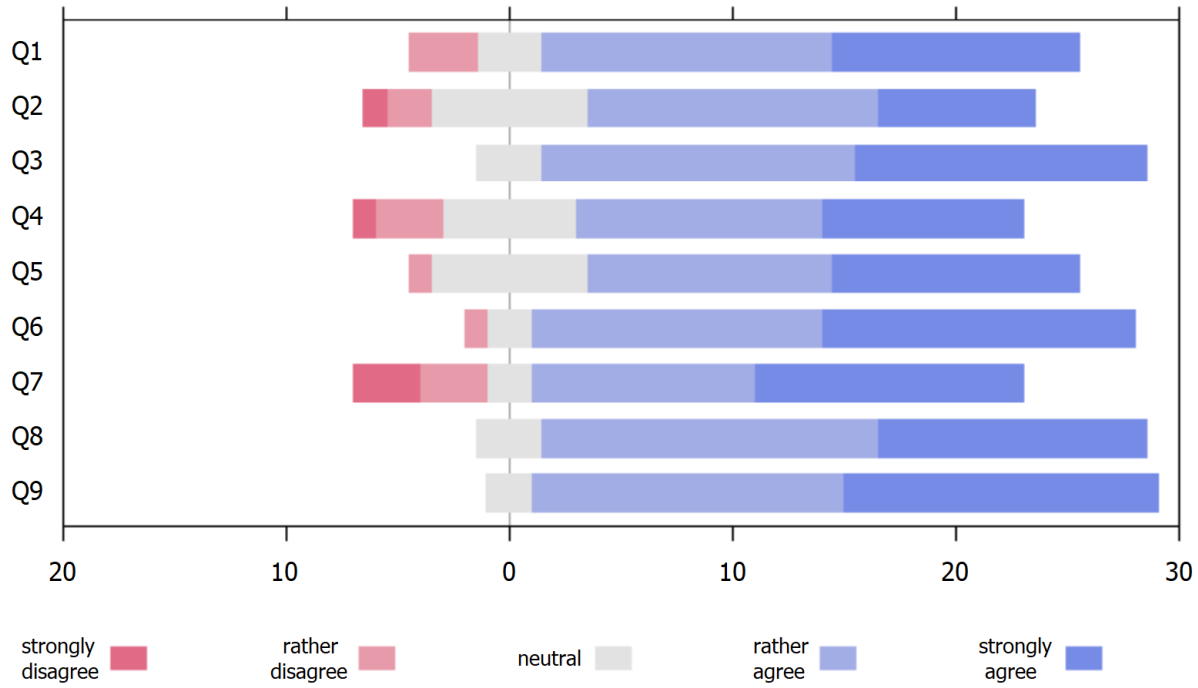

Figure 3. Feedback on the proposed customizations; the 2nd AR cycle

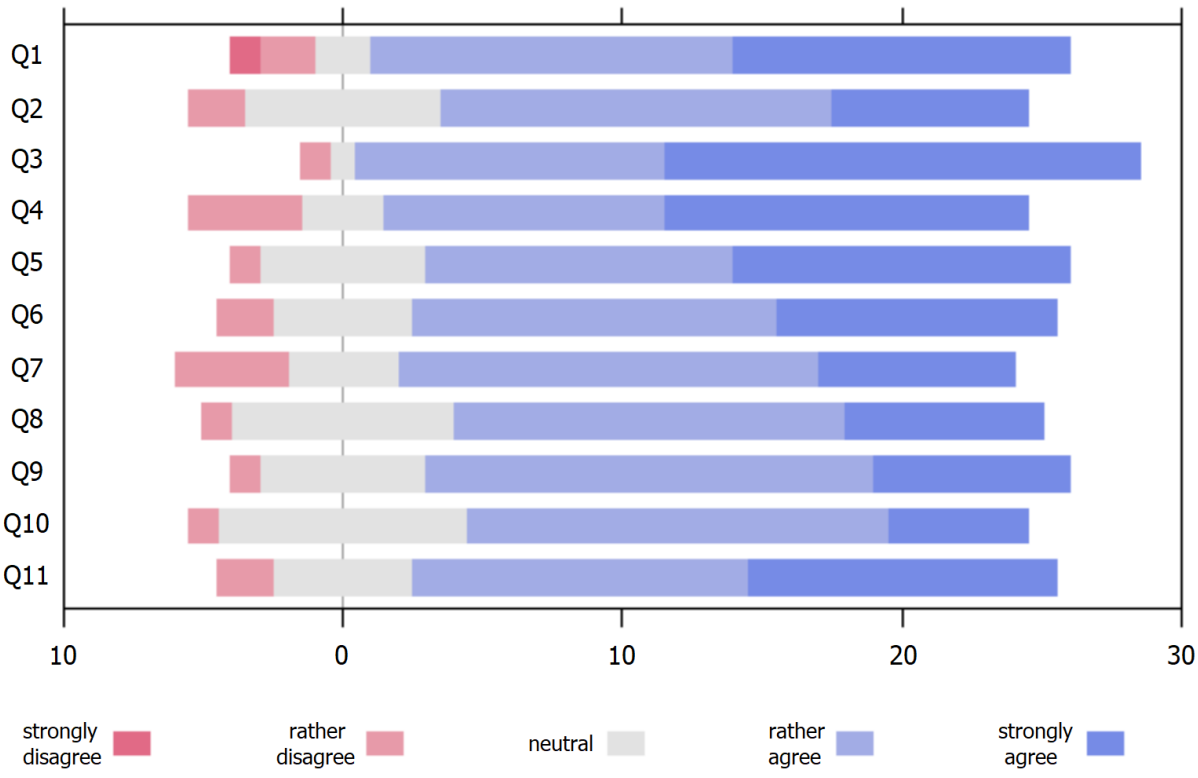

Figure 4. Feedback on the proposed customizations; the 3rd AR cycle