Received: 16 August 2021 | Revised: 14 July 2022 | Accepted: 27 July 2022

DOI: 10.1002/smr.2504

RESEARCH ARTICLE - EMPIRICAL

Journal of Software: Evolution and Process WILEY

Quality culture boosts agile transformation—Action research in a business-to-business software business

Matti Saarikallio | Pasi Tyrväinen

University of Jyväskylä, Jyväskylä, Finland

Correspondence
Matti Saarikallio, University of Jyväskylä, Jyväskylä, Finland.
Email: matti@saarikallio.net

Abstract

Agile methodologies are sometimes adopted, with the assumption that benefits can be attained by only using a set of best practices, which can sometimes work to a degree. In this paper, a case is discussed where a software-producing organization of seven teams achieved significant improvements. The goal of the research was to answer two questions: how an already agile organization could improve its performance further and what is the impact of promoting quality aspects? The questions were answered by implementing interventions based on prior literature and data emerging from semi-structured interviews. The context was an established business with a complex revenue stream structure, meaning the mix of various project/service/product based work rendered the adoption of agile methods a challenge. Action research comprising three rounds of interventions was conducted to improve the organization and its quality culture while enforcing code review practices. Interventions resulted in a significant improvement in quality, as measured by reported defects. Therefore, it is suggested that agile methods are not sufficient on their own to take software business forward unless a quality-focused culture is simultaneously achieved through a mindset change and organizational structures to enforce quality practices. The paper contributes to research on the managerial practices of software business and agile transformation by providing empirical support to introducing formal quality improvement to the agile mix as a method for practitioners to improve organizations with complex business models and multiple teams.

KEYWORDS
agile adoption, B2B, empirical, hybrid development methods, increment planning event, mixed business model, quality, revenue stream, scaled agile, team coordination

1 | INTRODUCTION AND BACKGROUND

1.1 | Introduction

Research on agile transformation often involves analyzing the context of a simple business model and a small team, with the assumption that quality improvements will appear. In this paper, this approach is problematized in three ways. The first is related to difficulties faced by multi-team organizations in fully obtaining the expected benefits.[1] The second is companies with complex business models attempting to use these tactics,

which originate from simpler contexts.[2] The third (and most interesting) is the suspicion that the quality of software produced after the introduction of agile practices does not necessarily improve as much as anticipated. The goal of this paper is to investigate these three themes and report on action research that resulted in measurable improvements.

In this introductory section, relevant background research is summarized to analyze the context, and a question about possible limitations of agile methods is raised. This forms the theoretical background, rationale, and aim for the selected action research interventions.

1.2 | Software business and business model

The context of the current paper is software business. One way to understand this context more clearly is through business models, which bring a structured understanding of how firms conduct business and create value.[3] Established B2B (business-to-business) firms differ in their business models from new or B2C (business-to-consumer) firms. For example, established B2B firms tend to have personal communications with customers, have ways of measuring worker productivity, organize people in a role-based structure, focus mostly on process automation, have a larger organizational structure, and have more complex revenue capturing structures (such as through service contracts).[4] The business model of an established B2B software business can be further investigated starting from its revenue stream structure.[5] This means that the source of revenue streams, reasons for the existence of those streams, and the method of converting value to revenue in each stream should be considered. This type of analysis will be employed to determine the details of the contextual landscape of the case firm in this paper. Further, software businesses vary in terms of customer expectations or value drivers (such as product quality, service support, delivery, supplier know-how, time-to-market, personal interaction, price, and process costs).[6] Value drivers are also influenced by business logic.[7] Therefore, in the context investigated in the current paper, it is important to examine specifically what B2B customers value. This paper focuses on an organization that operates within an established B2B market segment. Therefore, the value drivers empirically determined by Parry et al.[8] are chosen as a frame, and it is assumed that customers value the following as the most important factors: (1) software quality, (2) professionalism, and (3) understanding the customer.

Agile methods originate from many sources. For example, scrum was influenced by the thinking of Takeuchi and Nonaka on how to speed up product development.[9,10] Due to this history, in traditional agile methodology practice, example businesses are suspected to have an overrepresentation of a single team producing work related to a clearly defined single product line. While optimal for simplifying an ideal model, this can result in ignoring some aspects of the real world, which consists of a multiplicity of different types of business models, motivating further research into other types of software organizations.

1.3 | Agile methods and quality practices

Since the start of the current millennium, agile development methods have become the de facto tactics for software producing organizations. Agile methods are described as incremental, co-operative, straightforward, and adaptive.[11] Core to these methods are agile manifesto principles that highlight the importance of interactions, working software, customer collaboration, and responding to change.[12] While quality is one of the key value drivers of software industry, as previously mentioned, the role of quality is rarely explicated in agile transformations. Quality may be implicitly expected to result from agile methods, for instance, from shorter iterations, the practices of pair programming, or developing the personal craftsmanship of the developers, but there is no explicit focus on developing quality practices.

When the size of an agile organization increases, issues related to the scaling of agility become relevant. While there is no single definition of what constitutes a large-scale team, typical suggestions are six teams or more,[13,14] rendering the current research (with seven teams) a borderline case. Napoleão et al.[15] analyzed the knowledge sharing activities in agile development, noting that sharing information inside the team is the knowledge management step that is strongly present in agile organizations. However, this would require team stability and does not consider cross-team concerns.

It has also been stated that agile methods are bespoke in origin,[16,17] meaning that the focus is on small projects. Helander and Ulkuniemi[7] indicated that customer perceived value differs in the sense that customers of project firms tend to value deep customer relationships, while those in product businesses place greater value on assurances of quality. As there is an increasing interest in software-as-a-service, it should be noted that investment in software quality is often necessary in such business models.[18] Similarly, the Industry 4.0 concept emphasizes quality and safety and digitizing total quality management has been suggested as a way forward in the industry.[19]

Sfetsos and Stamelos[20] conducted a systematic review of quality, specifically in agile practices. They highlighted test-driven development and pair programming as agile practices that contribute to code quality. More traditional quality practices were not mentioned. For example, research on the impact of inspecting software (currently called code reviews) existed long before agile practices emerged. Software inspection and reviews have a long research history.[21] This stream appears to live its own life, and a recent review of modern code review practices[22] only mentions agile as the reason why modern code review practices have replaced traditional formal reviews.

The importance of quality is undisputed if the goal is to produce professional software. Moreover, it has been suggested that the amount of reworking required for software development is as high as 44% of the total work effort, with the amount of defect insertion being highest during the early phases.[23] It is also quite easy to infer that defects found early in the process are easier to fix than those found later. Further, there is evidence that the more code is reviewed, the higher the quality of software.[24] While there is a common consensus in the software industry that code review practices are important, the actual implementation of such practices in organizations reveals problems. According to recent survey,[25] 76% of people conducting code reviews do so less than once a week.

In other industries, the quality of production processes is managed by controlling and measuring aspects of safety, quality, delivery, inventory levels, and production.[26] For example, the lean and Six Sigma methodologies (and the so-called seven basic tools of quality) have been considered essential tools in many industries.[27] Adopting lean practices in software development has also been perceived as useful,[28] attempts have been made to adapt quality function deployment to the software context,[29] and combining lean startup and user centered design approaches.[30] While agile methods have adapted ideas from these fields, some aspects may have been overlooked. Moreover, it would appear that current guides[31] on agile practices do not consider inspection and reviewing as essential ingredients for creating quality software.

Another simple method for organizations to improve quality is through the use of checklists. These are used in various fields, from flying bomber aircraft to medical surgery.[32,33] Agile methods promote the use of coding standards written down in form of definition of done. However, it is quite rare to find an actual checklist in software teams. The weak implementation in industry has been suggested an important research direction.[34] Moreover, while the agility goal of teams self-organizing toward better coding standards is good, this might not happen automatically.

The term quality culture is sometimes seen as something separate from quality management practices.[35] This is maybe due to quality standards often seen as regulating organizational processes or the development of individual abilities. However, in order to drive any organizational improvements, be it practices or methods, addressing organizational culture cannot be avoided. Culture here means those shared assumptions that are considered valid by the group, and taught as the correct ways to see, think and feel about their work.[35,36]

While contemporary agile research might have an under-representation of studies focusing on quality, the question is whether this is true in industry practice. Based on a consulting agency report[37] on agile adoptions, it is striking that the only mentions about quality relate to test automation, automated static analysis as part of continuous integration, or valuing craftsmanship. Accordingly, it is suspected that quality in the context of industry agile transformations sometimes tends to be reduced to suggesting the use of automation tools. This could lead to poor managerial decisions and the omission of best practices from general software research when transforming to agile practices.

1.4 | Rationale and aim

Based on the literature, the limited focus on quality in agile transformations requires investigation. The case under study in this paper was identified as representing a software business with quite a complex business model, which could be affected by this. The business had adopted agile practices approximately 7 years before the current research and was still struggling to gain hoped benefits. Thus, the aim of the current study is to contribute to the research domain of software business quality improvement practices in the context of agile methodology, especially in the context of a specific less-researched type of software business—the established B2B mixed-business model. The value production of such an organization requires a focus on software quality, professionalism, and understanding of the customer.[8] There could be a reduced interest in agile transformations pertaining to quality improvement aspects. It is suspected that in some companies it is assumed that agile transformations somehow fix quality automatically (with shorter iterations equated with better quality[38]). Therefore, the goal is to investigate whether agile methods are sufficient and if improvement opportunities exist through the introduction of additional quality checks.

Based on the literature and the highlighted question about quality focus in agile adoption in current industry practice, two research questions were considered important: (1) how can an already agile organization improve its performance further and (2) what is the impact of promoting quality aspects.

2 | METHODOLOGY

2.1 | Case study and action research

A case study strategy focuses on understanding the dynamics present in a single setting.[39] Although multiple teams were investigated, they all share a similar context, rendering this more of a single setting. The level of analysis is not on a single team; it is on the organizational behavior and effectiveness of the whole business unit.

Information systems science uses both case studies and action research.[40] The research reported in the current paper is not purely observational, because a process improvement was actively initiated, guided, and followed-up by the author. Therefore, it is better described as a combination of a case study and action research.

Software development is an undertaking that resides in the complex domain. Accordingly, software-producing organizations tend to be complex and variable. As a methodology, the aim of action research is to understand a social situation and its change processes. The goal of action research is to analyze a complex social-organizational model (or complex human process) in a real environment.[41] Participation of the researcher in planning the changes and intervening in the social system to improve the organization's functioning was considered a suitable method to gain a deeper understanding of this case. The goal was to gain knowledge and explore the applicability of interventions aimed at improving the organization's effectiveness.

Action research has the goal of actually changing (rather than simply describing and explaining) the organization.[42] Following this methodology, the initial recommendations are founded upon a theoretical frame. The recommendations lead to actions that produce experimental changes in the organization, and the actions and their impacts are reported and discussed. Lewin[43] formulated the cycle of action research as follows: early mapping of the situation, planning for changes, taking action, and making observations. Further, contextual analysis of behavior in a social setting is also emphasized. In his taxonomy of research methods, Järvinen[44] recently recommended action research if the goal was to achieve utility and build a better system. This renders it a relevant approach for this study, because the goal is to improve the output of a software-producing organization.

The case organization is described in the next chapter. Thereafter, the mixed method data collection techniques are described as integral parts of the conducted action research process. Mixed methods were employed because there is a need to understand the context and meaning in a social setting. In addition, there is a need to provide quantitative evidence about the impact of any changes, making the study easier to replicate and possibly generalize in the future.[45,46]

2.2 | Case

The focus of investigation is the business unit of a publicly traded Northern European software company. The investigated unit provides services and products for the financial services industry. Although the mother corporation has over 1000 employees, the focus of this analysis is the unit serving the financial sector (approximately 100 employees).

Revenue sources of the chosen business unit emanate from its operations in the financial services software market, providing products and services for banking and loans, wealth management, fund management, asset management, and custody and settlement domains. It is noteworthy that the financial sector was one of the first industries to adopt computers on a large scale to optimize business processes, with up to 50% of banks' fixed capital already spent on computers 40 years ago.[47] Therefore, it is unsurprising that a large amount of source code and fairly comprehensive product offerings can be found in established players serving this sector.

The method of revenue capture varied substantially. There were maintenance fees from existing installations, license fees, hourly-based fees on customer-specific small features, and larger developments delivered as various forms of projects. Thus, there are service contracts, product license contracts, hourly-based work, and SaaS models used in parallel.

The reasons why revenue can be captured are from activating product features or creating new functionalities for customers, promising maintenance with agreed service levels. Some customers are offered specific people as capacity, while others are sold work through (sometimes capped) hourly contracts. It is also fairly common that a product feature required by multiple customers has a type of kick-back discount if other customers choose to use it, instead of providing exclusivity of features. Typically, a monthly service fee is increased by a certain percentage when more features are added to that customer's service. While customers vary substantially in terms of their product configurations and customer-specific tailoring, it has been possible to maintain a common source code core and database model (to a degree), making products more maintainable.

Based on a short revenue stream analysis, the revenue stream structure of the company is quite complex. Moreover, it is easy to understand that the complexity of operations is high in our case, which has caused large variations across the operative teams. Hence, the case business unit is a typical small B2B multi-offering organization with a complex mix of services and products.

The business unit has offices in two countries and three cities, meaning it has a more national (rather than global) focus specializing in a deep understanding of a country's specific needs. The focal business unit's employee headcount is approximately 100, of which approximately 50 work in software development teams. Additional people work in service, product, and project management functions, as well as for sales and consulting. The operational model was based on separating the functions of sales, service management, project management, and other administrative roles from the software delivery organization.

2.3 | Procedure and data collection

Action research followed the phases of diagnosing, action planning, action taking, evaluating, and specifying learning. The action research procedure was cyclical (Figure 1), and three interlinked interventions were performed. During each intervention, the organizational situation was diagnosed by collecting data in the form of interviews, supporting documents, and observations (1). After diagnosis, a plan was formulated based on theoretical predictions emerging from data and supporting literature (2). During action taking (3), further observations were made and informal

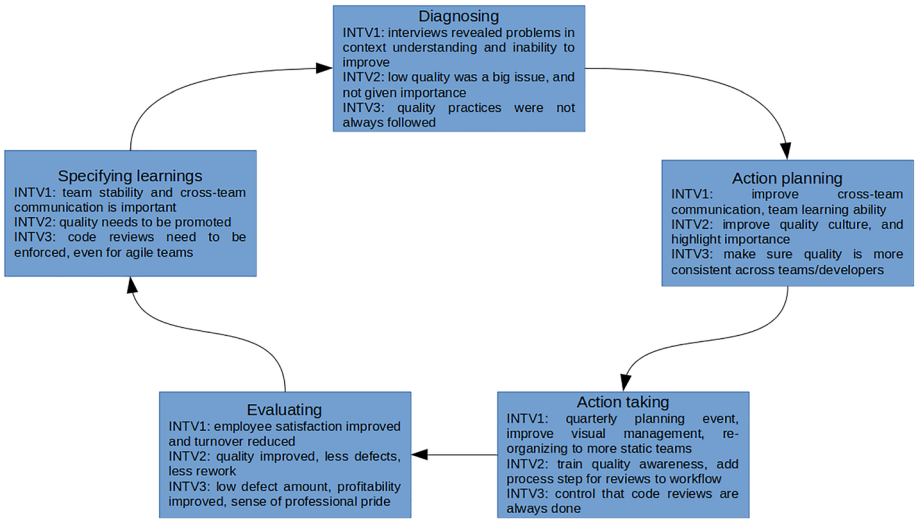

FIGURE 1 Action research cycle included three interventions.

TABLE 1 Initial interviews included 41 people

		Experience in business unit		
Role	**Informant count**	**<3 years**	**3–5 years**	**>5 years**
Programming	15	6	4	5
Architecture (technical)	6	1		5
Managing projects	5	1	2	2
Managing service	2	1	1	
Product management/analysis/business expertise	8		3	5
Scrum master/PO	3	1		2
Others (sales, QA)	2			2

interviews were conducted to understand the impact. Evaluation of the outcomes of actions (4) was conducted based on qualitative and quantitative data, and then reflected. Finally, the learnings were specified (5).

There were several data collection steps during the time window of this study, in accordance with the steps of action research. Data collection started at the end of Q1 2018 with interviews in the organization. The first goal was to quickly identify the most important problems (close timing also helped with avoiding cross-contamination of the interviews). Several changes were then introduced in the three action research iterations. Most of the initial data was relevant during all interventions, as the interventions targeted different aspects of the diagnosed issues. It should be noted that different from classical action research, the interventions were not fully sequential. Instead, they had an overlap in time with respect to the first and second interventions, because it was possible to drive some actions in parallel. Decisions needed for the actions (and learnings) were discussed in the business unit's management team.

2.4 | Diagnosing

As a preliminary diagnosis of the situation, 41 people (Table 1) in the organization were interviewed to gather insights into the current state of affairs in the business unit. The questions pertained to the biggest challenges faced by the organization, the reasons for these challenges, improvement possibilities, required actions to realize the improvements, and requesting advice on the most important focus areas (adapted from Watkins[48]). The collected interview data was analyzed by conducting inductive content analysis.[49] The interview notes were summarized and translated into English and then coded using open and 'in vivo' coding.[50] The list of similar text items included 354 lines that were grouped based on similarity to create categories. The number of occurrences of similar comments per category were then counted to elicit a sense of importance for each one. These categories were further organized into main categories, allowing the most important themes that emerged from the data to be identified and analyzed. Some emotionally charged or sensitive details were removed from the reported table due to ethical and confidentiality

considerations, although the data items were still counted in the analysis. Additionally, some of the organizational rituals, for example, scrum events, were observed. Further, secondary materials (such as documentation, and observation of meetings) were examined.

2.5 | Actions planned and taken

Action planning and action taking were implemented after the initial diagnosis, which were also based on observed early improvements. The first of the three interventions was initiated shortly after the interviews. Several changes were planned and implemented during the research time-span. The plans of action were based on the diagnosis of the situation and theoretical predictions. It was essential to gain enough organizational and management support to drive the changes through to ensure they would stick. Therefore, not all changes were attempted at once. This small-steps approach also helped in building the organization's change capability. Building general change capability was important to allow appropriate pacing and sequencing to give time for adaptation to the new situation.[51]

2.6 | Evaluating

Evaluation of the intervention outcomes was initially based on qualitative data such as observations and interviews. In addition, secondary documentation (such as management team meeting memos, architect meeting memos, and retrospective notes) was cross-checked.

While qualitative measures are good for in-depth understanding, a quantitative measure for the organization's ability to produce quality software was chosen to increase the reliability of the results. A good measure for this was the number of defects reported. Thus, evaluating the impact gained by actions taken was mainly based on measuring the number of defects reported during each quarter (3 calendar months). Data for the number of defects were obtained from the issue tracking system, which was the same system used for new functionality, defects, and tasks. The whole business unit was chosen as the unit of analysis; hence, the sum of all reported defects was obtained for each time interval. During the evaluation period, the effective head count remained almost flat (standard deviation of 2.05) with no trend. Accordingly, the data were not adjusted according to head count.

There were two defect count measures available from the issue tracking system: defects opened as internal (originating from anyone) and defects originating from customer's acceptance testing or other user reports (originating from customers). The conclusions were drawn from the customer-reported defects, because this measures the quality of the produced output more clearly. Similar measures have been adopted in previous research on product quality.[52] It should be noted that a defect here only included those issues that were suspected to require code changes. Typically, this was non-invoiceable work, meaning the practical relevance was high for the business. The measure did not include general support work (such as customer guidance or training).

Quantitative data were analyzed to ensure that observed changes in quality were relevant. This was evaluated by calculating the likelihood that results of changes could be explained by chance. This calculation was based on the assumption that the defect count followed a Poisson distribution, which is typical for integer count data. Null hypothesis: Changes in quarterly defects can be explained with random variation, meaning the process output quality did not improve and the before and after data points emanated from the same underlying distribution. Alternative hypothesis: quarterly defects reduced, meaning the process output has improved and before and after data points emanated from different distributions.

2.7 | Specifying learnings

As the interventions had the predicted impact, they were further solidified by making them enforced practices and the culturally expected ways of working. This was mainly accomplished by promoting good initial results in team presentations and explaining how this would constitute expected behavior in the future. The learnings were specified, communicated to participants, and maintained by updating documents. The common ways of working were also written down as living documents in the business unit's wiki.

3 | RESULTS

3.1 | Initial results leading to interventions

The short interpretative literature that was summarized earlier formed the main theoretical assumptions and motivation for the action research interventions and was complemented by themes emerging from the data. In this section, the qualitative interview data common for all interventions

are listed, then each intervention is described sequentially. The quantitative results are shown in the evaluation of Intervention 2, although they are also relevant for the diagnosis and evaluation of Intervention 3. Next, the categories and sub-categories emerging from the interviews are briefly listed. The grouping of the 354 coded sentences in the text data resulted in 7 main categories and 31 sub-categories (Table 2).

The first (1) main category was **cultural issues and instability** (103 comments were categorized here), which included the sub-categories of hurry ("temporary firefighting has become permanent"), turnover, negative cultural issues, instability ("a lot of people moved from team to team"), team spirit, resourcing, interruptions, role clarity, and positive cultural issues. The second (2) main category was labeled **vision** (79 comments), which included the sub-categories of product vision/management ("we need to clarify the story"), lack of leadership, context, and pricing/selling. The third (3) main category included comments relating to **process and quality** (67 comments) and sub-categories relating to process, testing, documentation, code reviews ("there is no control that the code review was done before it goes to testing"), scrum issues, refactoring, maintenance, and definition of done. The fourth (4) main category related to **product and quality** (40 comments), including sub-categories relating to technical complexity ("it's not possible to find a person who would know this all"), continuous integration, technical debt, and product quality. The fifth (5) main level theme was related to **project management** (28 comments), including sub-categories of administrative project management, bureaucracy, and project versus product. The sixth (6) main category was about **knowledge and customer** (35 comments) related comments, while the seventh (7) was related to other comments (such as personal advice given to the researcher about what to focus on and how to drive change).

After categorization, the issues were discussed within the organization and business unit's management team. Then, the highest priority improvement possibilities were selected based on business needs and the theoretical likelihood of success. It was decided that the most important were cultural issues and stability (including communication issues about vision) and quality issues (process and product). The other issues were not forgotten, although they were not the focus of improvements. A short summary of the interventions and motivations leading to them is provided in Table 3. Next, each step of the ensuing interventions is described in more detail.

3.2 | Diagnosis (intervention 1)

The interviews revealed that people had problems in understanding the context of their work. There was also a lack of product vision; it seemed that separation of teams prevented communication across boundaries, and there was a lack of communication across projects and teams. There was also a feeling of people "jumping around" and a lack of team stability. Apparently, some people had left the company due to organizational problems, communication issues, and rushed projects, creating a sense of hurry.

Reflecting on previous research literature, it was theorized that one of the reasons for the problems was that the scaling of agile methods was failing, as the size was seven teams. Additionally, it was suspected that the identified problem of moving people from team to team prevented effective team learning.

3.3 | Action plan (Intervention 1)

Action planning was conducted in collaboration with the organization. Partially based on the theories of agile scaling in the literature, it was predicted that improving communication, adding more visual workflow management, and increasing team stability would have positive impacts on the identified issues. The action plans were discussed within the organization's management team before taking action to ascertain support.

3.4 | Taking action (Intervention 1)

The first change was the introduction of a quarterly planning event that was adapted from the Scaled Agile Framework. This is suggested as a good choice when teams are struggling with information sharing and synchronization across teams.[53] The team structure was not changed for the first 3-month increment. Starting from the second increment, the team structure was changed toward business domain competence -based teams. Further, visual management was improved by unifying the visualization of issue workflows (kanban board), which were shown in daily scrum meetings. This constituted fine-tuning, since varying related practices were already in place in most teams.

3.5 | Evaluation (Intervention 1)

After the first intervention, the turnover problems reduced and remained less than the corporate average for the length of the study. The interviews revealed that the general levels of satisfaction increased and team stability remained quite good, which allowed the teams to bond more effectively. In addition, complaints about not understanding the context were reduced.

TABLE 2 Themes emerging from the interviews formed the basis of problem diagnosis

Main category	Category	Comments	Example codes
Cultural issues and instability	Hurry	46	Hurry, no free developers, uneven workload, impossible schedule promises, firefighting, overworked architects
	Turnover	12	Experienced people leaving, people leaving, cannot recruit new people, new people taking large shoes
	Negative culture	10	No ability for renewal, self-organizing was a red flag, blaming, separation between management and production
	Instability	8	Moving people around, borrowing people, people frustrated about throwing people around
	Team spirit	8	Team spirit, team does not feel shared responsibility, team member feel alone, spirit of working together
	Resourcing	7	Resourcing challenges, projects competing for same resources, steering groups do not make decisions only discussions
	Interruptions	6	Lot of interruptions, things are fragmented, frustration
	Role clarity	4	Lack of role clarity
	Positive culture	2	Great developers, good people who support each other and help
Vision	Product vision/ management	32	Lack of long-term planning, no product roadmap, too wide offering, I have not seen a project roadmap, doing what customer says
	Lack of leadership	27	Reluctancy to decide, following up on execution, people are seen as cost, team model needs to work better, no strategy
	Context	15	No context, information through rumors, no discussion between projects
	Pricing/selling	5	Selling non-profitably, no clear pricing model, expanding business beyond legacy
Process and quality	Process	27	Lack of professionalism, we take shortcuts all the time, no clear processes, sprints leaking, chaotic operations, homey feeling
	Testing	14	Quality assurance weak, no planned testing, regression tests are done if there is time, not enough automated tests
	Documentation	8	No documentation, very light document reviews, db documentation, manuals are not updated
	Code reviews	7	No code reviews, code reviews are difficult, Only 2 years of code reviews at all, no interest in quality
	Scrum issues	3	Scrum masters are in a difficult spot
	Refactoring	3	No refactoring, no interest in quality
	Maintenance	3	Maintenance issues, maintenance should be inside teams, maintenance not valued enough just fixing quickly
	Definition of Done	2	No definition of done, cross team reviews would be good
Project management	Administrative project management	24	Project management is administrative, poor planning, finalizing large projects is hard, spoon-feeding style, have not seen project plans
	Bureaucracy	2	Reputation of bureaucratic processes
	projects versus product	2	Product versus customer versus project
Knowledge and customer	Knowledge	18	Siloed knowhow, knowledge only in some peoples head, too few experts, new do not get up to speed, individual performance
	Customer	17	More customer facing people, good customer relationship, too much following customer's whistle, reactive mode
Product and quality	Technical complexity	12	Monolithic product, complex product, large/huge codebase, difficult to learn product
	Continuous integration	12	Too many code versions, breaking other branches, change logs are difficult to make, merging code across products and customers
	Technical debt	10	Too much tech debt, duplication, quality varies, bugs, maintenance is caused by poor quality
	Product quality	6	Lack of functionality, good package but poorly wrapped, tailored configurations per customer, not a real product
Other	Advice for researcher	4	Let us not try too much at once, listen to us and be critical, need a proactive convincing attitude, only the loudest get a voice

TABLE 3 Interventions had empirical and theoretical motivations

Stage	Time period	Motivation from interviews and data	Theoretical assumptions
Intervention 1: Improving communication, visual management, and team stability.	Q2-2018 → Q4-2019	Key people were leaving, complaints about lack of context, people felt they were "jumping" around, and context was unclear.	Scaling agile was failing, and team learning did not have time to happen.
Intervention 2: Promoting quality and recommending code reviews.	Q3-2018 → Q3-2019	Complaints about lack of testing and process and product quality issues.	Scrum was the main agile method used, and it does not promote quality well enough.
Intervention 3: Enforcing code reviews.	Q3-2019 → Q4-2020	Measuring amount of reviews showed it was not always done.	More code reviews improve quality.

3.6 | Learnings (Intervention 1)

The first intervention's results were as predicted, although they happened much slower than anticipated. The main learning was that in the context of this type of mixed business model in a borderline scaled agile organization, it is important to keep the teams fairly stable and ensure that there is a visualization of the workflow. Further, it is important to provide a mechanism for sharing information between teams via additional processes, such as increment planning events.

3.7 | Diagnosis (Intervention 2)

The diagnosis of the second intervention was based on the finding that both process and product quality were considered problems, with the theory suggesting that quality is not necessarily the main concern in scrum. In fact, an informant mentioned agile method as a reason (or excuse) for not focusing on quality. While interview data is subjective, this resulted in examining this issue further, revealing that quality was not afforded much importance. It was discussed further that there was a tendency to think quality was the concern of an individual developer's skill rather than something that could be improved at an organizational level. Interview data also included the comment "there is no definition of done." However, checking this fact revealed that there was one, although it was hard to find and was followed quite randomly. Moreover, the process did not contain a point where this could be enforced by the team. In the interviews, there were only a few discussions on code reviews, with most informants commenting that code reviews were not conducted. However, after checking, it was revealed that some people conducted these occasionally and that there was already a tool for this purpose.

The contract structure (ref. revenue stream analysis in case description) can worsen the situation with quality problems, as the fixes are often free. Unfortunately, there was no easy quantification of this issue due to the lack of consistent project reporting data; hence, the evidence is only based on the interviews. It was speculated that a long history of using a more traditional project-based plan-driven operational model had contributed to some of these quality issues, as the interview results about feelings of hurry might lead to a lowering of quality standards in some cases.

3.8 | Action plan (Intervention 2)

The plan of action for the second intervention was formed based on the following concept: increasing quality awareness in the organizational culture and highlighting the importance of quality practices would lead to fewer defects, improved morale, and increased profitability due to reduced warranty work.

3.9 | Taking action (Intervention 2)

Two forms of action were taken to implement the quality improvements. First, a cultural change was driven through informal team meetings and discussions. Lean thinking, prioritizing work in the order of safety, quality, inventory, productivity (+QDIP model[25]), and the idea that quality and efficiency are not opposites were also touched upon in these team discussions. Especially for junior team members, the idea of personal review checklists was recommended. The quarterly introduction to the increment planning event was also a good tool for vocalizing leadership support on the importance of quality. Further, one-on-one discussions with each individual developer and team meeting discussions were used to drive the initiative. The second form of improvement was related to the structure of workflow visualization. The kanban boards that were unified earlier for each team were redesigned to include an additional step between the development and testing phases. This step was named "in review," with

the instruction that it was to be used for indicating that the work item was in code review. Automatically, this created peer pressure toward actually conducting code reviews. Hence, it was no longer possible to skip the code review step without the developer making a conscious decision about moving the work item forward. Moreover, the issue management system would retain a record of this decision. At this point, the code review step was still only promoted as recommended practice, not one that should be enforced. The rationale for first promoting and not making it mandatory was to tackle possible resistance to change and drive the change though early adopters and thought leaders, instead of eliciting vocal opposition from laggards.

3.10 | Evaluation (Intervention 2)

Following the second intervention, people felt that quality was improving and that there were fewer defects. Informal interviews (especially with quality managers, testers, and architects) revealed that the amount of reworking (iterative multiple rounds of bug fixing) had reduced. The comments reflected that this was possibly due to stronger feelings of professional pride from the developers.

To improve the reliability of this finding and evaluate the impact of the intervention, a simple quantitative measure of reported software defects was employed. This revealed a good impact from the first intervention. Further, by simply training and promoting a quality culture, the defect count started to reduce (see Figure 2 and further results in the evaluation of Intervention 3). This started a cultural change in the organizations, and awareness of the importance of quality started to increase.

3.11 | Learnings (Intervention 2)

The main takeaway from Intervention 2 was that the organization needed to promote quality in addition to agile work management tactics. It was also understood that through quality promotion, a reduction of rework was starting to happen, which could have implications for profitability. It should be noted that without the stronger team workflow visualization being in place, it would not have been possible to increase the adoption of code review practice so easily.

3.12 | Diagnosis (Intervention 3)

The starting point for another intervention related to the realization from observations that quality practices were not always followed. Thus, further diagnosis was undertaken. At the end of 2018, a crude estimation of the amount of code reviews was conducted by checking the number of requests for reviews and how many were actually completed. It transpired that 63% were requested and only 48% were completed. It was also noted anecdotally that "it could take about a month to get your code reviewed." Thus, it seemed that quality practices (especially code reviews) were not always followed.

3.13 | Action plan (Intervention 3)

Although average quality had started to improve, variations in quality can lead to many problems. Accordingly, the next plan of action was to reduce variation across teams and developers and make quality more consistent.

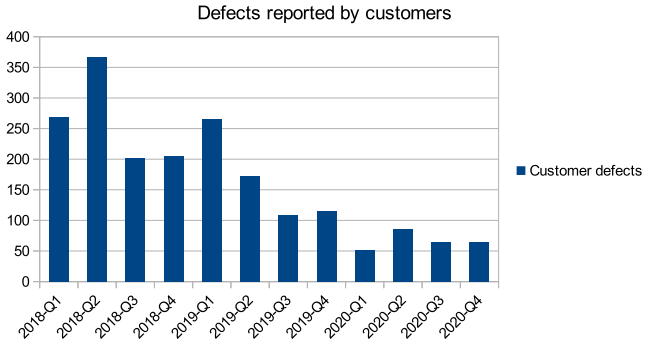

FIGURE 2 Quantitative results indicate quality improvement.

3.14 | Taking action (Intervention 3)

The goal of the third intervention was to improve the likelihood of code reviews happening, which was made possible by controlling that code reviews were always conducted. Two mechanisms were used to make this possible. First, it was communicated to the teams that the goal for 2019 was that "all code is reviewed." Second, the percentage of changes that were actually reviewed was calculated and communicated monthly. This metric of code review coverage increased further from 75% to 88% within a year from the start of the intervention. The remaining code that was not being reviewed mainly involved small fixes, which were excluded due to diminishing returns.

3.15 | Evaluation (Intervention 3)

Looking at the whole period of the three interventions, the third maintained quality improvements and improved the situation further. The number of defects reduced from an average of almost 300 in 2018 to an average of under 100 in 2020 (Figure 2). Assuming that defect data follows a Poisson distribution (lambda = 164), the likelihood that this reduction could be explained by chance was 0.0000. Hence, the null hypothesis could be safely rejected and the alternative hypothesis could be considered true, which supports the conclusion that quarterly defect amounts reduced. Additionally, there was a clear practical significance, which was observed from the reduction of non-invoiceable hours. A follow-up on the case business further showed that the new practices were still in place at the end of 2020, and the final months of the year set a record for the business unit's profitability. While this might have been partially attributable to circumstances (such as less traveling due to Covid-19), it could be speculated that focusing on quality had a positive impact on performance in the form of profitability improvements.

During 2020, an additional metric was introduced to the organization that measured the average cycle time from the start of development to the end of testing. This was possible after the stabilization of team workflows had been achieved. Based on this data, a reduction of cycle time was observed (all team averages per quarter in workdays: 34.6, 20.4, 23.5, and 14.7). Thus, it could be speculated that the increased quality also started to reduce cycle time. Since other interventions were conducted (such as highlighting slow-moving work in retrospectives), it is not absolutely clear to what degree cycle-time reduction was caused by quality improvement or other circumstances.

Some additional circumstantial evaluation was possible through yearly employee satisfaction surveys, and a positive trend of improving results continued throughout the research period. While the results could be attributed to many things, based on the free comments in the yearly employee survey data, it would appear that complaints about not understanding the context and feelings of hurry had reduced. There were also direct mentions about the increment planning event being a contributing factor. However, it can be speculated that increased quality and reduced bug fixing also had a positive effect on developer morale. This was a good result, because it has been suggested that increased employee happiness generally results in higher productivity.[54]

3.16 | Learnings (Intervention 3)

The main learning from the third intervention was that it is not sufficient to simply let teams self-organize and stumble upon quality practices. Further, code reviews need to be enforced, even for agile teams. Controlling for quality practices happening had a clear impact on the quality of the produced software in the investigated organization.

4 | DISCUSSION OF FINDINGS

This paper set out to explore three challenges of agile transformations: a multi-team organization that was having challenges in obtaining benefits from scrum method was investigated. The problems predicted by the literature were present, but it was possible to address them with interventions targeting communication structure and quality culture improvements. Additionally, it was speculated that companies with complex business models attempting to use agile tactics would have challenges. It was demonstrated that the complexity of the business model does seem to have a negative impact, but these issues can be addressed. For example, the increment planning event, although created for a different context, applies in this context as well. Initially, there was a suspicion that the quality of software produced after the agile practice introduction is not as high as the model is sometimes claimed to achieve. This was confirmed and demonstrated in a case organization, and fixing this problem required the interventions to focus more on boosting the quality culture and enforcing some quality practices, in addition to the prescription of agile tactics.

More specifically, two questions were answered. (1) The first was as follows: "How can an already agile organization improve its performance further?" Action research was conducted to improve team stability, cross-team communication, and the quality focus of the organizational culture. Further, code review practices were added into an organization that was already following agile practices. As a result, it was demonstrated that an organization that had earlier adopted agile practices was able to improve its performance. There was a significant reduction in reported defects,

and secondary results included improvements in employee satisfaction and profitability, which could be partially attributed to the introduced changes.

Compared with earlier research on the identified challenges found in agile transformations, one of three challenge areas identified by Laanti et al.,[38] namely, visibility and transparency, was strongly identified here, as it is comparable to our emerging category of vision-related issues. We found a rich amount of other categories of issues that emerged from the empirical diagnosis: cultural issues, team stability, vision and context understanding, process and quality, product and quality, project management, and knowledge and customer (see Table 2). From these, team stability is something that could be caused by repeated organizational changes, which paradoxically can sometimes be justified as a way toward strategic agility.[55] It seems that at least on the team level constant organizational structure changes can cause problems, preventing essential team bonding from happening. Good results were gained by stabilizing teams into business domain competence area-based teams. This goes somewhat against the consensus of agile thinking that a feature team is a preferable choice.[56,57] Maybe the business domain complexity and large size of the offered product are a factor.

The current research supports earlier studies on the problems of scaling agile development. Typical problems when scaling agile practices to multiple teams include testing and coordination issues,[58,59] which were found to be present. The interventions reported in this paper were able to address both of these through creating more stable teams and by adapting the Scaled Agile Framework's increment planning events. Thus, empirical support is given for the applicability of the planning event, even to business contexts where a clear product line or "value stream" is not easy to identify. Motivational benefits and increased shared contexts appeared to be the main wins. The novelty here is that although the existing contractual structures made it difficult to implement a product-line-oriented SaFE model, the adapted planning event still provided benefits. Incorporating such structured information-sharing events seems to be a good tool to provide a better contextual understanding.

(2) The second question for investigation was as follows: "What is the impact of promoting quality aspects?" The main portion of the action research focused on testing the impact of quality-promoting interventions. As a result, a substantial reduction in the amount of defects and an overall increase in business profitability, professional pride, and morale were observed. Therefore, this paper adds some empirical evidence that the promotion of quality aspects (especially enforcing code reviews) in addition to normal agile adoption is important for improving the performance of established software businesses. This latter result is in line with earlier reports with other improvement approaches when introducing reviews.[60] Nevertheless, the code review introduction created similar results in the agile context. Relating to professional pride, the correlation between employee (emotional and cognitive) engagement and quality culture has been suggested in other fields,[61] and there might be a mutually reinforcing relationship between them.

There has been an unfortunate trend in agile adoption of promoting a kind of revolution against processes, instead of truly integrating real industry needs.[62] While a revolutionary approach has value in being a working marketing gambit for promoting agility in general, it could be better to go beyond fighting the straw-man called waterfall and start integrating factors such as business model understanding and proven quality practices as integral parts of agile best practices. This paper provides a small critique of the agile methods movement and its ability to drive software business performance improvement all the way. The suggested fix hinted at by the reported evidence is to reintroduce a focus on quality as a best practice. The managerial implication is the recommendation that when attempting to gain the benefits of agile transformation, make sure that quality improvement practices (especially code reviews) are a key part of the agenda. It is a bit odd that the agile manifesto does not include code reviews as a suggested practice, because it seems to be an easy practice to implement and has real benefits. This omission could be because pair programming seemed like a better (similar) alternative at the time. Still, the practices have been suggested to have complementary benefits from the perspective of team knowledge creation.[63]

4.1 | Limitations and future research

It has been said that the fast pace of industry changes makes the validity of empirical research in software organizations limited.[64] As a mitigation to this general problem, this paper attempts to provide a rich contextual description to give an understanding of the circumstances present. Generalizing too much on these results should be avoided because action research can overstate the importance of the specific interventions,[65] but applying similar interventions in additional software-producing organizations would be interesting.

One limitation could be that the investigated business unit did not follow agile practices sufficiently, rendering it a less representative sample. It could be argued that some of the problems identified were related to not following scrum guidelines fully. The counterpoint to this is that when compared to the so-called scrum anti-patterns,[66] only a few of them were present in the diagnosed situation, and not all teams had the same issues. Moreover, it could be argued that the case of a poorly followed scrum is more typical in real life compared to an ideal model.

There was also a limitation related to validity, which is inherent in action research. Accordingly, it is not entirely clear which additional factors have an impact on the results. Although mixed method data were used, repeating the interventions in another company with similar characteristics of having adopted agile practices (although still having quality and profitability problems) would be interesting. This research has presented an empirical example of how to improve a software-producing organization. While the applicability to general software organizations is limited, there are many mixed-business-model organizations that might be less researched than purer business models. Further, it is not clear if the origin of the

identified problems was more related to the complexity of the business model or the scaled agile size. Therefore, further studies in similar contexts are required.

It has been suggested that the software process improvement (SPI) maturity of the organization has an impact on the quality.[60] One limitation was that we did not measure the SPI maturity with standard measures, although the reported data in Table 2 could have been used to speculate on this. Maturity increase might explain some part of the results. It would be good in future studies to control for SPI maturity.

Pair programming has been seen as valuable in creating shared mental models and backup behavior, in addition to improving performance in novel tasks.[67] Incorporating pair programming is not the easiest change, especially with the increase of remote working. Accordingly, it would be useful to evaluate whether similar benefits can be obtained by code review practices. Large companies have to contend with massive code bases. For example, Facebook seems to have adopted a compulsory code review practice as a necessity to enable continuous deployment.[68] Within really large companies, an interesting avenue for future research could be the use of AI to partially automate code review processes. This would require a substantial amount of training data, which can only be collected if the code reviews are conducted and documented appropriately. Thus, it is the opinion of the author that rigorous code reviews remain an essential part of any professional-level software organization.

5 | CONCLUSION

While the agile movement has fundamentally affected the way software is produced today, many are still struggling to achieve the promised benefits. This paper has outlined some issues that should be considered. It was highlighted that (re)introducing code reviews is an essential ingredient of success in software business, which should not be forgotten even in agile organizations. This might be especially true in established organizations.

The nature of qualitative research is to explore the investigated context, and definitive answers are not claimed. Regardless, the likelihood of the following conclusion seems higher: even for organizations that have used agile practices for a while, if the organization has a complex business model (B2B, mix of services/products) and is larger than six teams, problems with scaling the agile organization are likely. These problems can be improved by introducing more cross-team communication (e.g., increment planning events and stabilizing team structures) to allow for learning to happen and by promoting quality awareness. Further, if the code base is large and complex, benefits from agile testing practices are harder to implement. Therefore, introducing compulsory code reviews can be a recommended course of action to improve organizational performance.

This paper contributes to industry practice by demonstrating some improvements in operational excellence that are possible in the context of a mixed-offering established software-producing organization. For the practicing manager, first analyzing the context starting from revenue streams, finding the categories of issues present, measuring cycle times and defect rates, and boosting agile transformation by creating a quality-aware organization, are the principal suggestions.

DATA AVAILABILITY STATEMENT

Research data are not shared.

ORCID

Matti Saarikallio https://orcid.org/0000-0003-2813-9146

REFERENCES

1. Kalenda M, Hyna P, Rossi B. Scaling agile in large organizations: practices, challenges, and success factors. *J Softw Evol Process*. 2018;30(10):e1954. doi:10.1002/smr.1954
2. Kruchten P. Contextualizing agile software development. *J Softw Evol Process*. 2013;25(4):351-361. doi:10.1002/smr.572
3. Zott C, Amit R, Massa L. The business model: recent developments and future research. *J Manag*. 2011;374(4):1019-1042. doi:10.1177/0149206311406265
4. Vanhala E, Saarikallio M. Business model elements in different types of organization in software business. *Int J Comput Inf Syst Ind Manag Appl*. 2015;7:139-150.
5. Takeuchi H, Nonaka I. The new product development game. *Harv Bus Rev*. 1986;64(1):137-146.
6. Ulaga W. Capturing value creation in business relationships: a customer perspective. *Ind Mark Manag*. 2003;32(8):677-693. doi:10.1016/j.indmarman.2003.06.008
7. Helander N, Ulkuniemi P. Customer perceived value in the software business. *J High Technol Managem Res*. 2012;23(1):26-35. doi:10.1016/j.hitech.2012.03.003
8. Parry S, Rowley J, Jones R, Kupiec-Teahan B. Customer-perceived value in business-to-business relationships: a study of software customers. *J Mark Manag*. 2012;28(7–8):887-911. doi:10.1080/0267257X.2012.698637
9. Abrahamsson P, Warsta J, Siponen MT, Ronkainen J. New directions on agile methods: a comparative analysis. In 25th International Conference on Software Engineering, 2003. Proceedings. IEEE; 2003;244-254.
10. Saarikallio M, Tyrväinen P. Following the money: revenue stream constituents in case of within-firm variation. In international conference of software business. Springer, Cham; 2004:88-99 doi:10.1007/978-3-319-08738-2_7

11. Abrahamsson P, Salo O, Ronkainen J. Warsta J. Agile software development methods: review and analysis, VTT publication 478 2002. Espoo, Finland; 107.
12. Beck K, Beedle M, Van Bennekum A et al. Manifesto for agile software development. 2001.
13. Dikert K, Paasivaara M, Lassenius C. Challenges and success factors for large-scale agile transformations: a systematic literature review. *J Syst Softw*. 2016;119:87-108. doi:10.1016/j.jss.2016.06.013
14. Abrar MF, Khan MS, Ali S, et al. 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
15. Napoleão BM, de Souza ÉF, Ruiz GA, Felizardo KR, Meinerz GV, Vijaykumar NL. Synthesizing research on knowledge management and agile software development using the meta-ethnography method. *J Syst Softw*. 2021;178:110973. doi:10.1016/j.jss.2021.110973
16. Karlstrom D, Runeson P. Combining agile methods with stage-gate project management. *IEEE Software*. 2005;22(3):43-49. doi:10.1109/MS.2005.59
17. Dzamashvili Fogelström N, Gorschek T, Svahnberg M, Olsson P. The impact of agile principles on market-driven software product development. *J Softw Maintenance Evol Res Pract*. 2010;22(1):53-80. doi:10.1002/spip.420
18. Choudhary V. Comparison of software quality under perpetual licensing and software as a service. *J Manag Inform Syst*. 2007;24(2):141-165. doi:10.2753/MIS0742-1222240206
19. Jamkhaneh HB, Shahin A, Parkouhi SV, Shahin R. The new concept of quality in the digital era: a human resource empowerment perspective. *TQM J*. 2022 Emerald;34(1):125-144. doi:10.1108/TQM-01-2021-0030
20. Sfetsos P, Stamelos I. Empirical studies on quality in agile practices: a systematic literature review. In 2010 seventh international conference on the quality of information and communications technology. IEEE; 2010:44-53.
21. Fagan ME. Design and code inspections to reduce errors in programs. *IBM Syst J*. 1979;15(3):219-248. doi:10.1147/sj.153.0182
22. Davila N, Nunes I. A systematic literature review and taxonomy of modern code review. *J Syst Softw*. 2021;110951. doi:10.1016/j.jss.2021.110951
23. Brykczynski B, Meeson R, Wheeler DA. Software inspection: eliminating software defects. Institute for Defense Analyses. Alexandria, Virginia; 1994.
24. McIntosh S, Kamei Y, Adams B, Hassan AE. An empirical study of the impact of modern code review practices on software quality. *Emp Softw Eng*. 2015;21(5):2146-2189.
25. Dosea M, Sant'Anna C, Oliveira Y, Colaco JM. A survey of software code review practices in Brazil 2020. arXiv e-prints, arXiv-2007.
26. Protzman C, Whiton F, Kerpchar J, et al. Getting ready to implement lean system. In: *The Lean Practitioner's Field Book*. Productivity Press; 2018. doi:10.4324/9781315373843
27. Magar VM, Shinde VB. Application of 7 quality control (7 QC) tools for continuous improvement of manufacturing processes. *Int J Eng Res Gen Sci*. 2014;2(4):364-371.
28. Poppendieck M. Implementing lean software development: from concept to cash. Pearson. Boston; 2006. ISBN 0-321-43738-1.
29. Jamkhaneh HB, Shahin R, Shahin A, ArabYarmohammadi M. CMMS software quality function deployment based on maintenance objectives: a framework for software selection process. *Int J Prod Quality Manag*. 2021;32(4):413-439. doi:10.1504/IJPQM.2021.114260
30. Zorzetti M, Signoretti I, Salerno L, Marczak S, Bastos R. Improving agile software development using user-centered design and lean startup. *Inf Softw Technol*. 2022;141:106718. doi:10.1016/j.infsof.2021.106718
31. Schwaber K, Sutherland J The scrum guide 2020. Accessed online 2022. https://scrumguides.orgdocs/scrumguide/v2020/2020-Scrum-Guide-US.pdf
32. Emerton M, Panesar SS, Forrest K. Safer surgery: how a checklist can make orthopaedic surgery safer. *Orthopaedics Trauma*. 2009;23(5):377-380. PMID: ISSN 1877-1327.
33. Bohm R, Not flying by the book: slow adoption of checklists and procedures in WW2 aviation. San Diego; 2013.
34. Kollanus S, Koskinen J. Survey of software inspection research. *Open Softw Eng J*. 2009;3(1):1-34. doi:10.2174/1874107X00903010015
35. Ehlers UD. Understanding quality culture. *Qual Assurance Educ*. Emerald. 2009;17(4):343-363. doi:10.1108/09684880910992322
36. Schein EH. *Organizational Culture and Leadership*. 2nd ed. Jossey-Bass.
37. Hyde P. 10 ways your agile adoption will fail. Gartner; 2019.
38. Laanti M, Salo O, Abrahamsson P. Agile methods rapidly replacing traditional methods at Nokia: a survey of opinions on agile transformation. *Inform Softw Technol*. 2011 Elsevier;53(3):276-290. doi:10.1016/j.infsof.2010.11.010
39. Runeson P, Höst M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw Eng*. 2009;14(2):131-164. doi:10.1007/s10664-008-9102-8
40. Eisenhardt KM. Building theories from case study research. *Acad Manage Rev*. 1989;14(4):532-550. doi:10.2307/258557
41. Baskerville RL. Investigating information systems with action research. *Commun Assoc Inform Syst*. 1999;2(1):19. doi:10.17705/1CAIS.00219
42. Coghlan D, Brannick T. *Doing action research in your own organization*. Sage; 2001.
43. Lewin K. Frontiers in group dynamics: II. channels of group life; social planning and action research. *Hum Relat*. 1947;1(2):143-153. doi:10.1177/001872674700100201
44. Järvinen P. Improving guidelines and developing a taxonomy of methodologies for research in information systems. Jyväskylä. JYU dissertations; 2021.
45. Creswell JW, Klassen AC, Plano Clark VL, Smith KC. *Best Practices for Mixed Methods Research in the Health Sciences*. Maryland: National Institutes of Health 2013;2011:541-545.
46. Kaplan B, Duchon D. Combining qualitative and quantitative methods in information systems research: a case study. *MIS Q*. 1988;12(4):571-586. doi:10.2307/249133
47. Franke RH. Technological revolution and productivity decline: computer introduction in the financial industry. *Technol Forecasting Soc Change*. 1987;31(2):143-154. doi:10.1016/0040-1625(87)90046-1
48. Watkins M. *The first 90 days – critical success strategies for new leaders at all levels*. Harvard Business School Press; 2003:45-46.
49. Elo S, Kyngäs H. The qualitative content analysis process. *J Adv Nurs*. 2008;62(1):107-115. doi:10.1111/j.1365-2648.2007.04569.x
50. Strauss A, Corbin J. *Basics of qualitative research*. 2nded. Thousand Oaks, CA: Sage; 1998.
51. Meyer CB, Stensaker IG. Developing capacity for change. *J Change Manag*. 2006;6(2):217-231. doi:10.1080/14697010600693731
52. Agrawal M, Chari K. Software effort, quality, and cycle time: a study of CMM level 5 projects. *IEEE Trans Softw Eng*. 2007;33(3):145-156. doi:10.1109/TSE.2007.29

53. Paasivaara M. Adopting SAFe to scale agile in a globally distributed organization. In 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE). IEEE; 2017:36-40.
54. Graziotin D, Fagerholm F, Wang X, Abrahamsson P. What happens when software developers are (un)happy. *J Syst Softw*. 2018;140:32-47. doi:10.1016/j.jss.2018.02.041
55. Lamberg JA, Lubinaitė S, Ojala J, Tikkanen H. The curse of agility: the Nokia Corporation and the loss of market dominance in mobile phones, 2003–2013. *Business History*. 2021;63(4):574-605. doi:10.1080/00076791.2019.1593964
56. Olsson HH, Bosch J, Alahyari H. Towards R&D as innovation experiment systems: a framework for moving beyond agile software development. In Proceedings of the IASTED; 2013:798-805.
57. Larman C, Vodde B. Practices for scaling lean and agile development: large, Multisite and Offshore Projects with Large-Scale Scrum.
58. Petersen K, Wohlin C. The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Softw Eng*. 2010;15(6):654-693. doi:10.1007/s10664-010-9136-6
59. Berntzen M, Stray V, Moe NB. Coordination strategies: managing inter-team coordination challenges in large-scale agile. In: Gregory P, Lassenius C, Wang X, Kruchten P, eds. *Agile Processes in Software Engineering and Extreme Programming. XP 2021*. Vol. 419. Cham: Lecture notes in business information processing. Springer; 2021.
60. Pries-Heje J, Johansen J. SPI manifesto. European system & software process improvement and innovation. 2010. Accessed online 2021: eurospi.net.
61. Dewanto A, Febrina SS, Wardhani V. The importance of nurses' cognitive and emotional engagement in developing hospital quality culture. *Enferm Clin*. 2020;30:97-101. doi:10.1016/j.enfcli.2020.06.022
62. Ebert C, Paasivaara M. Scaling agile. *IEEE Software*. 2017;34(6):98-103. doi:10.1109/MS.2017.4121226
63. Spohrer K, Kude T, Schmidt CT, Heinzl A. Knowledge creation in information systems development teams: the role of pair programming and peer code review. ECIS, 2013;CR,213.
64. Fernández DM, Passoth JH. Empirical software engineering: from discipline to interdiscipline. *J Syst Softw*. 2019;148:170-179. doi:10.1016/j.jss.2018.11.019
65. Myers MD. *Qualitative research in business and management*. Sage; 2019:77-77.
66. Eloranta VP, Koskimies K, Mikkonen T. Exploring ScrumBut—an empirical study of scrum anti-patterns. *Inf Softw Technol*. 2016;74:194-203. doi:10.1016/j.infsof.2015.12.003
67. Kude T, Mithas S, Schmidt CT, Heinzl A. How pair programming influences team performance: the role of backup behavior, shared mental models, and task novelty. *Inf Syst Res*. 2019;30(4):1145-1163. doi:10.1287/isre.2019.0856
68. Distefano D, Fähndrich M, Logozzo F, O'Hearn PW. Scaling static analyses at Facebook. Commun. ACM; 2019;62,8 (August 2019):62-70. doi:10.1145/3338112

How to cite this article: Saarikallio M, Tyrväinen P. Quality culture boosts agile transformation—Action research in a business-to-business software business. *J Softw Evol Proc*. 2022;e2504. doi:10.1002/smr.2504