



The end of agile as we know it

Downloaded from: <https://research.chalmers.se>, 2026-10-28 08:32 UTC

Citation for the original published paper (version of record):

Bosch, J., Olsson, H. (2027). The end of agile as we know it. Journal of Systems and Software, 243.
<http://dx.doi.org/10.1016/j.jss.2026.113100>

N.B. When citing this work, cite the original published paper.



The end of agile as we know it[☆]

Jan Bosch^{a,b}, Helena Holmström Olsson^{c,*,*}

^a Chalmers University of Technology, Gothenburg, Sweden

^b Eindhoven University of Technology, Eindhoven, The Netherlands

^c Malmö University, Malmö, Sweden

ARTICLE INFO

Keywords:

Agile software development
Agile practices
DevOps
Software-intensive embedded systems
Challenges
New paradigm

ABSTRACT

Context: Ever since the Agile Manifesto was introduced in 2001, the agile principles have supported software development teams in continuous delivery of valuable software. With its focus on customer collaboration, and with a promise to help organizations respond to change, it is fair to say that the agile paradigm changed software development and that it, over the years, has become the method of choice worldwide.

Objectives: However, despite more than two decades of efforts being spent on adopting and scaling agile practices, companies still do not experience the benefits they had expected. In this paper, we study the challenges companies experience and why existing agile practices fall short in supporting development of large-scale software-intensive embedded systems.

Methods: We present a reflective synthesis across a decade of collaborative research within a large research consortium. We do so by synthesizing findings across a selected set of our own prior studies into higher-order interpretations of the challenges that software-intensive systems companies face when implementing, adopting and scaling agile practices. In addition to reflecting on our previous research, we report on a workshop series conducted in 2024 in which we refined and extended the challenges we identified and where we explored what a new paradigm could potentially look like.

Results: First, we identify the key challenges that companies in the software-intensive embedded systems domain experience in relation to agile practices. Second, we empirically derive the key principles that a novel, next generation product development paradigm needs to satisfy. Third, we present a conceptual framework, i.e., RADICAL, a strawman proposal for what a such new paradigm could potentially look like.

Conclusion: We conclude that for the case companies involved in our research, there is a mismatch between the original agile principles and the reality of product development they face. Hence, there is a need for a new paradigm that supports not only development, but also scaling and monetization of software-intensive embedded systems.

1. Introduction

Ever since its introduction in the early 2000s, the interest in, the use of, and the controversies surrounding agile development have only increased (Dingsøyr et al., 2012). Often, the change towards agile development is described as a paradigm shift that fundamentally changed software development. Although the challenge of how to organize software development to deliver faster, better, and cheaper solutions is not new (Dybå and Dingsøyr, 2008), the agile manifesto enforced a shift from plan-based and process-oriented development methods towards more light-weight and responsive development methods emphasizing speed, incremental cycles and short iterations for engaging with customers. In research, the introduction, adoption and evolution

of agile practices have been carefully studied and there are prominent papers exploring e.g., agile transformation and learnings from these (Baskerville et al., 2011), agile adoption and evolution over time (Mathiassen and Sandberg, 2021; Yürüm et al., 2018), agile maturity and how to assess this (Turetken et al., 2017), organizational culture and how culture affects the adoption and use of agile practices (Iivari and Iivari, 2011), the challenges involved in scaling of agile practices in systems involving mechatronics components (Eklund and Berger, 2017), and overall business agility and how to achieve this (Mathiassen and Pries-Heje, 2006; Olsson and Bosch, 2024b). As can be seen in these papers, as well as in numerous other papers studying agile, the scope of agile practices is wide as it covers everything from the organization of cross-functional teams to the mechanisms to scale agile practices to achieve

[☆] This article is part of a Special issue entitled: 'Agile Reloaded' published in The Journal of Systems & Software.

^{*} Corresponding author.

E-mail address: helena.holmstrom.olsson@mau.se (H.H. Olsson).

overall business agility. Hence, the topic keeps attracting attention as researchers as well as practitioners seek to understand, benefit and improve the principles.

During the last two decades, the use of agile practices have spread from the small software teams for which they were initially intended, to large-scale development of embedded systems and Internet of Things (IoT) applications (Hoda et al., 2018). To achieve this, frameworks such as SAFe (Paasivaara et al., 2012) and Scrum-of-Scrums (AlMutairi and Qureshi, 2015) have been introduced to scale agile practices to fit for the development of complex and large projects involving multiple teams, multiple stakeholders and multiple development sites.

However, despite years and years of efforts going into adoption and scaling of agile practices, companies still do not experience the benefits they had expected and that made them adopt agile practices in the first place. While the methods, tools and processes keep increasing in number, and while the advocates of agile keep arguing for the benefits, companies struggle with adopting and adjusting to the paradigm and challenges remain despite their hard efforts and investments. In research, there are numerous studies reporting on the many challenges that hinder effective adoption as well as factors that support and facilitate adoption (see e.g., Misra et al. (2009), Ruk et al. (2019), Chandra Misra et al. (2010)).

In this paper, we present a reflective synthesis across more than a decade of collaborative research within a 16-company research consortium. We do so by synthesizing findings across a selected set of our own prior studies, involving 14 of the 16 company members, into higher-order interpretations of the challenges that software-intensive systems companies face when implementing, adopting and scaling agile practices. The companies use agile practices, many of them have adopted SAFe and they aim for DevOps ways-of-working. Despite years of efforts being spent on adopting agile practices, our research shows that for these companies, there is a mismatch between the original agile principles and the reality of product development they face. Hence, there is a need for a new paradigm that supports not only development, but scaling and monetization of software-intensive embedded systems. Throughout this paper, we use the term “product development” to denote the development and continuous evolution of software-intensive embedded systems, i.e., products that combine mechanical, electronic, software and AI components, as distinct from the development of pure-software products. As a second data source, we report on an on-site and online workshop series that was conducted to refine and extend our previous insights and specifically, for generating ideas for what a future paradigm could potentially look like.

The research questions we explore are the following:

- RQ1: *What are the key challenges of agile that software-intensive systems companies experience?*
- RQ2: *What are the key principles that a next generation product development paradigm, suitable for software-intensive embedded systems companies, should satisfy?*
- RQ3: *What would an approach to address these challenges and satisfy these principles entail?*

The contribution of this paper is threefold. First, we identify the key challenges that companies in the software-intensive embedded systems domain experience in relation to agile practices. Second, we empirically derive the key principles that a novel, next generation product development paradigm needs to satisfy. We intentionally broaden the scope from agile practices to product development as the evidence shows that the problem is not within agile teams, but rather with scaling beyond teams. Third, we present RADICAL, a strawman proposal for what a such new paradigm could look like. The findings we synthesize originate in a series of case studies that we conducted with the case companies during a 14 year-period of collaborative research. While our prior studies were multi-case studies, the present paper is a reflective synthesis of the findings in our prior work. Our strawman proposal is

a conceptual contribution and a design artifact in which we outline the dimensions, the focus areas and the tools that we believe are critical for a new product development paradigm. RADICAL represents what we believe could succeed today's agile practices to better support the reality of product development in software-intensive embedded systems companies. Also, we provide three illustrative examples of how elements of the proposed framework are emerging and being applied in practice. The contribution we present is a higher-level integration that was not necessarily visible within any single prior case study, but that emerged from systematically relating findings from across our prior studies.

From a practical perspective, the findings in the paper validate, extend and challenge the existing academic body of knowledge in the following ways. First, the key challenges that are identified (Section 4) provide practitioners with a shared vocabulary and a diagnostic instrument in that the challenges are formulated at a level at which engineers, architects and managers in software-intensive embedded systems companies recognize their own situation and can be used by a company to assess which of these challenges apply to them and where their current agile adoption is constrained. Second, the seven key principles (Section 5) function as design requirements in that rather than prescribing a process, they give companies a basis on which to evaluate their existing ways-of-working and any candidate framework they consider adopting. Third, RADICAL (Section 6) offers a concrete, if deliberately provisional, target state, with focus areas and associated tools identified per dimension, which companies can adopt incrementally rather than as an “all-or-nothing” transformation. In Section 6.3, we show elements of how this is already emerging in practice. Hence, our findings *validate* prior work reporting on challenges in large-scale agile adoption and transformation, they *extend* existing work by showing that in the software-intensive embedded systems domain the challenges are not confined to software R&D but extend into business models, systems engineering and organizational governance, and they *challenge* the prevailing assumption that these difficulties can be resolved by scaling frameworks such as SAFe or Scrum-of-Scrums, since our data indicate that the limitation originates in the assumptions of the agile paradigm itself rather than in its incomplete implementation.

The paper is structured as follows. In Section 2, we review literature on agile practices and we present our interpretation of agile. In Section 3, we describe the research method, the research process and the case companies involved in our research. In Section 4, we identify the key challenges that the companies we studied experience in relation to agile practices. In Section 5, we present the key principles that a new development paradigm should follow. Section 6 presents RADICAL, a conceptual framework and our strawman proposal for what we believe could be the next paradigm after agile. In Section 7, we discuss our results in relation to previous research. In Section 8, we discuss threats to validity. Finally, in Section 9, we conclude the paper and outline future research opportunities.

2. Background

In this section, we review literature on agile practices and we present our interpretation of agile. The aim of the section is to provide an overview of existing research and to conceptualize agile as experienced by practitioners in the companies we studied.

2.1. The evolution of agile

Few terms have received as much attention as the term ‘agile’ and few concepts, if any, have been applied as broadly across disciplines. While its origin, and intended focus, was software development, the term agile has spread far beyond the software domain and is used also in e.g., banking (Birkinshaw, 2019), human resource management (McMackin and Heffernan, 2021) and construction (Mohamed and Moselhi, 2019) to mention a few. With the agile manifesto being published

in 2001, the emphasis shifted from plan-driven processes and heavy documentation towards more responsive ways-of-working with a strong customer focus (Fowler et al., 2001) and ever since, the agile principles have been adopted across domains and it has spread from small, co-located teams developing bespoke software to large-scale development of systems intended for a mass-market (Eklund et al., 2014; Eklund and Berger, 2017).

In Baskerville et al. (2011) the authors present a future in which there is no distinction between agile and plan-driven development methods. Instead, they foresee a scenario in which the dual goal of agility, i.e., responsiveness and frequent value delivery, as well as alignment of people and tasks, is achieved through a diversity of means, and by using agile, plan-driven, and other types of method components. In Mathiassen and Sandberg (2021), the authors present a retrospective reflection on how development practices based on new technology paradigms and development practices based on previous paradigms may eventually complement each other and become closely intertwined. In Hoda et al. (2018), the authors outline the evolution of agile and show how agile practices have shifted from focusing on development methods in the late 90s, to focusing on large-scale agile and embedded systems in the mid 2000s, to most recently being applied also in development of emergent technologies. As part of this evolution, the authors recognize the rise of DevOps and continuous practices.

The evolution of agile is reflected also in several studies focusing on aspects such as agile adoption, e.g. Kurapati et al. (2012), F. Tripp and Armstrong (2018), agile transformation e.g., Laanti et al. (2011), Russo (2021), challenges and success factors in large-scale agile transformation initiatives e.g., Paasivaara et al. (2018), Dikert et al. (2016), responsiveness in agile development of products intended to a mass-market (Olsson et al., 2013) and business agility e.g., Olsson and Bosch (2024b), Mathiassen and Pries-Heje (2006). As can be seen in many of the published papers, there has been an increasing focus on the scaling of agile and the introduction of these practices in the software-intensive embedded systems domain, Figalist et al. (2019), Eklund et al. (2014), Olsson and Bosch (2014). Recently, the literature on agile was complemented with papers recognizing BizDevOps as an emerging trend that seeks to accelerate integration of functions and the scaling of agile cycles even further. In addition, there are several systematic literature reviews on agile development that provide an overview and summary of its evolution (Edison et al., 2021; Kišš and Rossi, 2018; Dikert et al., 2016; Perkusich et al., 2020). In our own previous work (Olsson et al., 2012), we identify a pattern that companies follow as they evolve towards continuous practices. This pattern involves five steps starting with traditional development and with the last step being R&D as an experiment system.

The notion of agile is not only used in software development, but also in other disciplines such as systems engineering. Authors such as Dove (2005), Turner (2007) and Douglass (2015) discuss the application of agile principles to systems engineering where the resulting system includes mechanical, electronic and software components. Although agile systems engineering has brought valuable contributions to a field that traditionally has been characterized by waterfall processes, the practices, in our experience with the companies that we collaborate with, suffer from two main limitations, i.e. their dependence on requirements and their focus on a life-cycle approach where there is a start of production when development stops.

In addition to the papers referred to above, there are many additional papers that outline the evolution of agile and that summarize how agile development, since its inception in the late 1990s, has come to dominate the latter half of the past 50 years of software engineering. In our own previous research, we covered several areas and topics related to agile practices when studying e.g., the early introduction of agile practices (Olsson et al., 2012; Olsson and Bosch, 2014), the scaling of these practices (Eklund et al., 2014; Eklund and Berger, 2017; Martini and Bosch, 2019), the transition towards continuous practices (Bosch, 2014; Dakkak et al., 2022; Lwakatere et al., 2016),

the specifics of large-scale systems development and the implications of agile practices (Olsson et al., 2013), the increasing importance of data and the shift towards data-driven development practices (Liu et al., 2021; Issa Mattos et al., 2023; Holmström Olsson and Bosch, 2024; Fabijan et al., 2018a,b) and the digital transformation enabled by software, data and AI technologies (Bosch and Olsson, 2021; Olsson and Bosch, 2020b; Figalist et al., 2022; Olsson and Bosch, 2025).

2.2. Definitions and our conceptualization of agile

Ever since the manifesto was articulated, practitioners and researchers have been trying to explicate agility and its different facets. As recognized in Dingsøyr et al. (2012), agility entails the ability to rapidly and flexibly create and respond to change. Over the years, numerous definitions have been proposed as attempts to capture the multiple aspects of agility and what can be achieved when using agile practices. In Williams and Cockburn (2003), agile development is described as being “*about feedback and change*”, In Conboy (Conboy, 2009), agility is the readiness “*to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment*”. In essence, agile development, and as reflected in its definitions, suggest a light methodology that promotes manoeuvrability and speed of response (Williams and Cockburn, 2003).

The purpose of this paper is not to explore which definition of agile might be the most comprehensive one. There are already rigorous studies that do this and over the years, we have seen several systematic reviews in which the term agile, and its associated characteristics, are studied in detail (see e.g., Dingsøyr et al. (2012), Al-Saqqa et al. (2020)). However, it is important to present and frame agile as we understand it and as we will refer to it in the remainder of this paper. In our experience, there are on the one hand, precisely defined agile frameworks such as e.g., SFA that some organizations try to adopt and follow to the letter. On the other hand, organizations that are well versed in agile have, over the years, developed an internalized definition of agile that includes everything that is new and useful, including continuous integration, continuous test, continuous deployment, A/B testing, DataOps, AIOps, etc. Hence, we focus our discussion on the common denominator that most practitioners in the companies we studied seem to agree on.

In Fig. 1, we present our conceptualization of agile. Our conceptualization is based on how practitioners in the case companies we studied understand agile, how they implement and use it and how it is currently helping their teams and their businesses to shorten development and deployment cycles. Hence, the conceptualization in Fig. 1 is our own illustration of how practitioners that we work closely with have communicated their perceptions and experiences of agile practices to us.

The intention with the figure is to provide a structuring of the scope of agile as we view it and as we use it in this paper. In the figure, we show how agile development practices start in the scope of teams that adopt and use agile methods. Over time, and as a first step to broaden the scope of these practices, continuous integration (CI) and test practices are adopted, meaning that the testing and quality assurance organization adhere to agile cycles. As the most mature stage, the organization adopts DevOps practices to allow for agile, and continuous, deployment of new software functionality. At this stage, agile practices move beyond the development organization as also the customer is involved in end-to-end deployment of software. With end-to-end, we refer to a company where every function involved in developing, deploying and evolving products uses agile practices. This is needed to create not only shorter cycles within development but to also deploy new functionality and create the associated business value. The N/A areas in the figure illustrate approaches that may be theoretically possible, but that are unrealistic in practice. For instance,

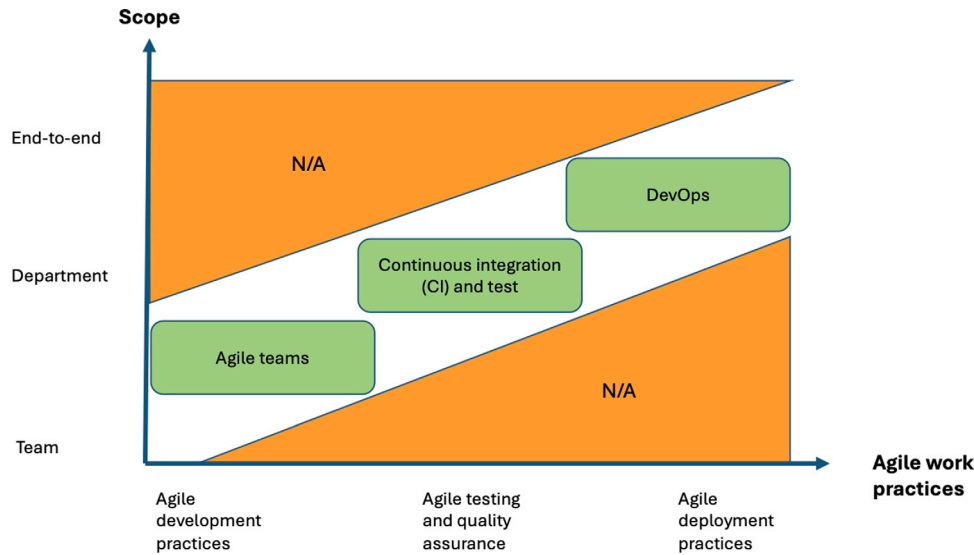


Fig. 1. Conceptualization of agile as experienced by practitioners in the case companies we studied.

end-to-end agile practices without quality assurance and deployment make little sense. Similarly, agile deployment at the team level without involving the rest of the organization is typically infeasible. It should be noted that the intention of Fig. 1 is not to outline the adoption process of agile development practices as a straight-line progression. We acknowledge that it is more complicated than that and that companies go through iterations, reorganizations and contingency plans during this process. Instead, the intention is to provide an illustration of the scope of agile practices as we see it being implemented in practice and as communicated by practitioners we work with.

3. Research method

In this section, we describe the research approach we adopted, the context in which our research was conducted, the case companies involved and how data was synthesized and analyzed. The aim of the section is to explain the research process and how results were derived.

In this study, we employ both reflective and longitudinal research approaches. The research reported in this paper is *reflective*, i.e., it revisits and synthesizes findings from prior studies of phenomena that have already occurred and been published and produces higher-order interpretations across them. The material on which the synthesis draws is *longitudinal*, i.e., it originates in a continuous research engagement with companies in the software-intensive systems domain covering a 14 year-period. This continuity is what gives the material its temporal depth and allows us to observe how the adoption of agile practices, and the difficulties associated with these developed and were experienced in these companies over more than a decade. In this sense the longitudinal character belongs to the underlying research engagement rather than to the analytical approach adopted in this paper. The workshop series conducted in 2024 constitutes an additional data collection effort carried out specifically for this paper to *refine and extend* the synthesized findings from our prior work. Also, it generates the material from which we derive the principles that are presented in Section 5. Below, we detail the two empirical components of this paper:

1. *Reflective synthesis of our previous research*: Insights that we accumulated during a 14 year-period of collaborative case study research in the case companies outlined in Table 1. In this paper, we present a retrospective reflective synthesis across more than a decade of collaborative research within a 16-company research consortium. We do so by synthesizing findings across a selected

Table 1

Descriptions of the case companies we worked with throughout our 14-year long collaboration.

Case company	Description
A	A company manufacturing trucks, buses, and construction equipment as well as a supplier of marine systems.
B	A food packaging and processing company.
C	A company manufacturing network cameras, access control, and network audio devices for the security and surveillance industries.
D	A company developing automation and digitalization solutions for process and manufacturing industries, intelligent infrastructure for buildings, and distributed energy systems.
E	A company manufacturing pumps and electronic motors.
F	A company developing software and systems in the areas of connectivity, security, mobility solutions, and AI.
G	A networking and telecommunications company.
H	A manufacturer of vehicles.
I	A company developing software for autonomous drive.
J	A supplier of green, safe, and reliable power control solutions.
K	A company offering navigational information, operations planning tools, flight planning products, and software.
L	An aerospace and defense company.
M	A provider of transport solutions, including trucks and buses.
N	A company providing high-assurance cybersecurity solutions.

- set of our own prior studies into higher-order interpretations of the challenges that software-intensive systems companies face when implementing, adopting and scaling agile practices.
2. *An on-site and online workshop series*: A more recent cross-company workshop series that was organized during 2024 to

Table 2

Data sources throughout our 14-years research collaboration with the case companies.

Data source	Description
More than 600 meetings (company workshops)	Meetings and workshop sessions at case company sites. These were documented resulting in a large volume of meeting minutes that reflect what was said, what was decided and what key stakeholders prioritized and supported.
More than 900 research presentations	Presentations at company workshops, company events and at workshops and conferences where we reported our research.
More than 600 documents (company presentations)	Presentations and documents provided by the company representatives as part of e.g., workshops where company participants presented their on-going work, their current practices and the challenges they faced.
More than 1500 informal interactions (discussions)	Discussions about challenges and opportunities with developers, architects, data scientists, product managers, senior leaders and other roles and functions in the companies.
More than 1000 formal interactions (interviews)	Formal interviews to explore challenges, opportunities and how to change and improve current ways of working with key stakeholders in the organizations.

refine and extend our previous insights and specifically, to identify the key principles required for a novel product development paradigm as well for generating ideas for what such paradigm could potentially look like.

3.1. Reflective synthesis of our previous research

In this paper, we present a retrospective reflective synthesis built on findings from our previous publications and the rich experiences and insights we accumulated during our 14-year collaboration with the companies. Using continuously expanding data sources from year 2011 and onwards, we have worked closely with other researchers and with practitioners to develop multiple publications that cover a broad range of topics that all relate closely to the experiences with agile practices. As recognized in [Cruzes et al. \(2015\)](#) most case studies in software engineering research are single-case or few-case studies, with large sample comparative studies still being seldom. The result is that knowledge about the phenomena of software engineering practices, methods, and techniques are spread over a myriad of diverse studies. In this paper, and similarly to [Mathiassen and Sandberg \(2021\)](#), we therefore aim to bring together rich case study material created by different researchers, contexts and study designs, with the overall intent to allow for a wider generalization than from single cases.

Our previous findings are derived from interview studies, observations, workshop notes, project artifacts and conversations that accumulated across many studies whose primary purpose was something else but where elements of agile practices, and the challenges with these, were always present. The data sources cover the implementation and use of agile practices over the considered period and the publications offer rich analyses of the challenges the companies experienced, potential solutions to these and the overall context of digital transformation in which the adoption of agile is only one part. As such, researchers and practitioners have established a well-functioning relationship in which insiders and outsiders have worked as a close collaborative team ([Mathiassen and Sandberg, 2013](#)) with both parties bringing their specific expertise to facilitate ongoing data collection, analysis and subsequent publication. Our reliance on multiple data sources throughout the research collaboration ([Table 2](#)) has supported triangulation ([Yin, 1994](#)) and helped us achieve higher reliability of our findings. While scrutinizing presentations, meeting minutes, other textual documents and informal discussions have been important in our ongoing research collaboration, the most important data sources have been direct involvement, participatory observations and formal assessment and research interviews.

It should be noted that the longitudinal research engagements allow us to not only explore what has happened, but also to explore what alternative ways-of-working could potentially look like going forward. The long-term research relationship has built trust between researchers and company representatives and hence, we have had the opportunity to engage in multiple projects, with multiple roles and with multiple functions in the companies. This allows for what we believe is a rich understanding of the many ways in which agile practices are perceived by practitioners.

All our previous research engagements were conducted as multi-case studies ([Dobson, 2001](#); [Runeson and Höst, 2009](#); [Maxwell, 2012](#)). Throughout our research, data was triangulated from multiple sources as suggested in [Yin \(2003\)](#). We conducted semi-structured interviews, we did participant observations, we reviewed documents (e.g., project reports and archival records) and we organized workshops at company sites. Data collection occurred in every research sprint using established protocols and guidelines to ensure consistency ([Runeson and Höst, 2009](#)).

3.2. Research context

For more than a decade, and as part of a larger research collaboration, we have had the privilege to work closely with multiple companies in the software-intensive embedded systems domain. The companies represent different industry domains and they are world class within their respective domain. The research is organized into four overarching research themes with a theme leader and several research projects in each theme. Our mission is to help the companies accelerate the adoption of new technologies and new ways-of-working and to support them in the digital transformation process that they are part of.

The research is organized into six-months sprints and for every sprint the member companies evaluate the research projects to decide on their continuation. This allows for continuous validation of results, the possibility for industry to steer the research direction towards the most prioritized areas of interest and the opportunity for researchers to engage in high-priority projects. While the set-up in the different projects might vary, the typical data collection methods are interviews, workshops, focus groups, observations, experiments and different types of interventions that allow the researchers to engage in solving real-world problems together with the company representatives who engage in the research projects. During each research sprint, the research results are continuously shared and validated with the companies to encourage frequent feedback and ensure accuracy of findings. This means that the findings we share in this paper, as well as the findings

in our previous publications, have been reviewed and approved by the company representatives.

During the last 14 years of our research, we have seen how agile methods were introduced in the companies, how they were adopted in the software teams and how the companies have been, and still are, trying to scale these methods beyond their software organizations to involve functions such as mechanics and electronics. Throughout our research, we engaged with the companies in case study research using workshops and interviews as our primary techniques for data collection. In addition, we engaged in seminars, events and in different types of company events to further strengthen our relationship to the companies and to learn even more about the opportunities and challenges they face. In this paper, the insights we accumulated provide the foundation for our understanding of agile practices and the challenges the case companies experience when using these.

3.3. Case companies

All case companies are members of a large research collaboration. When we started the collaboration in 2011, we had four member companies representing the automotive, the telecommunications and the surveillance domains. Over the years, the collaboration has grown and new member companies have continuously joined. Today, we have a total of 16 companies representing domains such as e.g., automotive, telecommunications, manufacturing, security, healthcare, logistics, surveillance, mobility and connectivity. All case companies use agile practices, all of them have continuous integration (CI) and test practices in place and several of them have adopted DevOps (Lwakatare et al., 2016; Dakkak et al., 2022). In this paper, we report on research that involves 14 of these companies. For more information please visit www.software-center.se.

Throughout our research, we defined our research topics and research questions in close collaboration with the member companies. The collaboration set-up is designed so that research projects are approved only if there are multiple companies interested as this allows for effective knowledge exchange between the companies and multi-case study research with the opportunity for generalization across cases for the researchers. Hence, the selection of case companies for a research study is based on interest from the member companies. They join a certain project to explore a certain topic that they feel are high priority at a specific point in time. This means that the companies involved in a project are the companies that, at that time, are the most relevant to study as they experience challenges, they need solutions and they are eager to share their perspectives with us and with other companies to improve the current situation. For each topic we study, we use input from all the member companies to generate results that are applicable and relevant for all involved companies and hopefully also beyond these.

In parallel with being responsive to the member companies, we also ensure that we together identify information-rich use cases that are closely aligned with our research objectives. This is known as purposive sampling (Campbell et al., 2020; Etikan et al., 2016) and as an approach that is recommended in qualitative case study research when the goal is to gain an in-depth understanding of a phenomenon rather than to obtain a statistically representative sample. In the early phase of each study, we therefore always discuss what use cases to bring in to ensure relevance and the ability to collect detailed empirical material from the participating companies. The case companies involved in the research represent critical instances where the studied phenomenon, i.e., the use and the scaling of agile practices, is mature and highly developed and thus particularly suitable for examining underlying challenges, concerns and issues that have emerged over time and as a consequence of adopting these practices. Also, the cases are practically relevant in that they are embedded in organizations that are representative of the broader population in terms of size, domain and technological

environment which supports the opportunities we have for generalization of our findings. Furthermore, the cases provide variation along dimensions such as domain, size, market position etc., allowing us to explore how the phenomenon unfolds under different contextual configurations and thereby enhancing the transferability of the results. It should be noted however, that the value of the research is not only the sum of the individual papers, but more than that. The true value is the rich understanding of the evolving context in which the case studies are conducted, the dynamic nature of the larger digital transformation processes and the ability to help companies stay competitive. In our experience, this larger context requires continuous adaptation to new contexts, skills and competencies. Hence, to add departments, units, or teams to our sample is a strategic approach to stay relevant and to keep our sampling strategy intact. The industry people who were added, i.e., people who joined the projects, were carefully identified by our key contacts at each company as they are the only ones who can ensure that the right expertise participates in a specific study. This cannot be done as effectively by researchers as the rapid change in practice, the organizational transformations and the constant dynamic in competencies are better understood in the companies themselves.

If looking at the maturity of agile practices, company G is by far the most advanced one with well-established DevOps practices since many years. The company had been using agile practices since early 2005 and the transition towards more continuous practices was a natural, and minor, step. Still, due to the size of the company the maturity levels might vary as is the case also in all other companies. A common situation is that some units are more advanced than others in using agile and DevOps practices. Although this is due to many reasons, a key reason is the type of technology the unit is responsible for. The further away from software you get the more difficult it becomes to employ agile practices.

In our research, we engaged primarily with the Swedish sites as many of the companies are headquartered in Sweden. However, company B has a site in Italy with which we collaborated, company D has its headquarter in Germany and we engaged with people there rather than the Swedish subsidiaries, company E and J are Danish companies to which we traveled on a regular basis during these years. Typically, our research is conducted in collaboration with software development organizations, data scientist organizations and people in architect roles but we have also had the opportunity to engage with product management, software process improvement organizations, management and innovation teams. The roles reflect the many different perspectives we cover in our previous papers and these go far beyond the topic of agile development. However, although many of our previous studies focus on other topics, there is often process and ways-of-working dimension where agile and DevOps are key elements.

In Table 1, we provide a summary description of the companies that provide the empirical basis for the results presented in this paper. As shown in the table, we engaged with 14 of the 16 member companies in the research consortium as the last two are companies that joined the research consortium very recently. The companies are anonymized and due to confidentiality we do not detail specific products, systems or the number of people involved in development of these. However, all case companies are large enterprises according to the OECD (Organisation for Economic Co-operation and Development) definition i.e., they employ more than 250 people.¹

All case companies are large international companies operating across the world and spanning multiple sites. Hence, our research covers primarily the Swedish sites of the companies, specific units within these and in most cases a selected set of teams in these units. While this could be viewed as a limitation, we believe that with the strong support we have from the companies and the opportunity to

¹ <https://www.oecd.org/en/data/indicators/enterprises-by-business-size.html>

engage in highly collaborative research involving experts that represent competence, skills and knowledge that goes far beyond their immediate team and unit, our results are valuable at an individual, a team and also a unit level. In the specific case of agile practices, the challenges we report on goes across individuals, teams and units as they are integral to the overall development approach in the companies. It should be noted that the challenges we identify in Section 4 are experienced by a majority, or all, companies. Hence, they apply broadly across the case companies.

3.4. Data synthesis and analysis

In the following subsections, we outline the two components of the study and we detail the design, the empirical data we collected and the analysis of this data.

3.4.1. Research component 1: Reflective synthesis of our previous research

This paper retrospectively synthesizes insights from a cumulative body of 14 years of case study research across multiple companies. What we present is a reflective synthesis across a decade of collaborative research within a large research collaboration involving multiple companies in the software-intensive domain. In our analysis, the findings from our prior studies are translated into higher-order interpretations of the challenges that the companies face when implementing, adopting and scaling agile practices.

As the basis for our analysis, we selected prior publications focusing on agile practices, continuous practices, large-scale systems development, data-driven practices and digital transformation. Building on overall thematic synthesis principles as presented in Cruzes et al. (2015) and inspired by the step-wise process as outlined in Clarke and Braun (2017), we treated the findings from our earlier papers as first-order material and systematically categorized, compared and combined them (the process is detailed in the Appendix). The contribution is a higher-level integration that was not necessarily visible within any single study but that emerged from systematically relating findings across prior studies.

In our synthesis, individual studies highlight different aspects of the phenomenon of study and were combined into a comprehensive storyline focusing on the challenges with introducing, adopting and scaling agile practices. Our thematic analysis unfolded as follows and in similar to what is outlined in Clarke and Braun (2017);

1. *Familiarization*: For each paper, we engaged in repeated reading to acquaint ourselves with the empirical contexts and original interpretations, thereby establishing familiarity with the material as a whole. As part of this process, we extracted the key reported findings in each paper.
2. *Generating initial codes*: As a second step, we worked through the papers more systematically. At this step, we identified the key challenges that could be associated with agile practices. For example, in Eklund et al. (2014) the key challenges that we identified relate to (1) *safety and certification requirements that are incompatible with iterative delivery*, (2), *agile frameworks designed for software do not transfer to hardware-dependent or embedded contexts*, and (3) *deployed embedded products cannot be updated rapidly, making iterative hypothesis validation with real users impractical*. In Olsson and Bosch (2014), the key challenges that were identified relate to (1) *difficulty measuring adoption maturity when terminology is ambiguous*, (2) *agile software teams operate at a different cadence than sales, procurement, legal, and hardware departments*, and (3) *customer-facing teams are not equipped to handle continuously changing products*. We listed the challenges we identified in all papers to get an overview of the experiences that were reported by practitioners.

3. *Constructing themes*: We examined the challenges we had listed and started clustering those that were related to each other, identifying patterns and proposing themes that would reflect the research questions we had defined. We organized conceptually related codes across papers. During this step, the different challenges were grouped together into high-level categories, e.g., *definitions and interpretations of agile*, *business aspects of agile*, *systems engineering challenges*, *organizational and cultural challenges* etc., to then be further reviewed and organized into more focused categories. This was an iterative process resulting in a structure with four overall themes of key challenges with a number of sub-categories in each theme.
4. *Reviewing themes*: We went through the themes we had proposed to ensure a coherent pattern and whether some codes should be moved, split, or merged. Also, we together discussed whether the themes we had identified indeed covered the understanding we have accumulated during the long-term collaboration and the insights we collected that go far beyond what is published in a specific paper.
5. *Defining and naming themes*: We went through the proposed themes, ensured the focus of each theme and decided on an informative label for each theme.
6. *Writing up*: In the write-up, we detailed the challenges that we had identified, provided examples and brought it all together. The resulting analysis presents a cross-case synthesis that brings together findings from multiple studies into a higher-level conceptual account of agile practices and the challenges that practitioners in software-intensive systems companies associated with these.

Although our collective research efforts involve more than 200 publications, we selected a limited set of papers as the basis for the synthesis. Below, we summarize the selected papers according to the following topics: *agile practices*, *continuous practices*, *large-scale systems development*, *data practices and value modeling and digital transformation* and we provide the corresponding references at the end of each subsection.

- In the papers on *agile practices*, we explored how to combine the planning and achieved predictability associated with mechanical manufacturing with more agile and dynamic planning, how to manage interdependencies between system components, how to allocate resources and ensure expertise in cross-functional teams and how to determine the optimal release frequencies for the different technologies that are part of an embedded system. These papers Olsson and Bosch (2024b, 2014), Eklund et al. (2014) are critical for the understanding of early introduction and also for later scaling of agile practices.
- In the papers on *continuous practices*, we studied how to move towards agile cycles also for test and deployment activities in a software-intensive embedded system, how to introduce and benefit from agile practices in operation functions such as customer support, how to ensure customer value in frequent delivery of software functionality, how to make use of data from products in the field to better understand customer value, how to increase responsiveness to customers in large-scale development and how to manage the balance between empowered teams and business goals in a large-scale development context. These papers Bosch (2014), Olsson et al. (2012), Dakkak et al. (2022), Stahl et al. (2017), Mårtensson et al. (2019), Backlund et al. (2014), Lwakatare et al. (2016) are critical for the understanding of evolving of agile practices and for steps that go beyond individual teams e.g., DevOps.
- In the papers on *large-scale systems development*, we explored how to define customer value in development of large-scale systems, how to achieve both scale and responsiveness and the trade-offs between these, the challenge of technical debt in large-scale

systems development and the emerging trend of empowered and autonomous teams. These papers [Olsson and Bosch \(2018a\)](#), [Olsson et al. \(2013\)](#), [Martini and Bosch \(2019\)](#), [Olsson Holmström and Bosch \(2016\)](#), [Olsson and Bosch \(2018b\)](#) are critical for the understanding of how to scale agile practices effectively in large organizations.

- In the papers on *data-driven practices*, we discussed how to conduct experimentation in safety-critical systems, how to manage and mature data practices in large-scale development of software-intensive systems, how to identify, agree upon and model customer value and how to align team efforts and business goals in development of software-intensive systems. These papers [Liu et al. \(2021\)](#), [Issa Mattos et al. \(2023\)](#), [Olsson and Bosch \(2024a\)](#), [Munappy et al. \(2022\)](#), [Olsson and Bosch \(2020a\)](#), [Holmström Olsson and Bosch \(2024\)](#), [Fabijan et al. \(2018a,b\)](#) are critical for the understanding of how agile teams increasingly work with data from products in the field as part of product improvements and feature development.
- Finally, in the papers on *digital transformation*, we explored the steps that companies evolve through when transitioning towards digital companies, how existing agile practices are complemented with DevOps, DataOps and MLOps to enable continuous development and deployment of digital technologies, what the novel approaches are that product managers can employ when having access to DevOps, the potential of data and artificial intelligence (AI) and the emerging concept of digital ecosystems. These papers [Bosch and Olsson \(2021\)](#), [Olsson and Bosch \(2020b\)](#), [Figaliet et al. \(2022\)](#), [Olsson and Bosch \(2025\)](#) are critical for the understanding of the overall digital transformation process that the companies experience and where agile ways-of-working is only one part and where software is complemented with data and AI.

Taken together, these selected papers provide insights that we accumulated during a 14 year-period of collaborative case study research in the case companies. The synthesis of these papers provides a holistic view by consolidating previous experiences into a structured understanding of why agile practices are challenging for companies in the software-intensive systems industry.

Throughout our engagements with the case companies, we used multiple data sources. Our reliance on multiple data sources supported triangulation of data, helped us minimize misunderstandings and helped us achieve higher reliability of our findings ([Yin, 2003](#); [Patton, 1999](#)). In [Table 2](#), we summarize the different data sources that we used over the years. It should be noted that the numbers are very conservative numbers and that they reflect 14 years of close collaboration. Also, while the unit of analysis in our prior work was typically teams and units/departments at the different companies the generalizations allowed for contributions beyond these. As an example, the stairway model that we present in [Olsson et al. \(2012\)](#) in which we outline how companies evolve from Traditional - Agile - Continuous Integration - Continuous Deployment - Data driven practices was derived from research conducted with R&D teams and units in multiple companies representing multiple industry domains. The contribution however, has been applied not only at team and unit levels but also to accelerate the evolution of the entire company and the larger industrial context of which it is part.

3.4.2. An on-site and online workshop series

In addition to the reflective synthesis of prior publications, we include findings from a more recent *cross-company workshop series* conducted between February 13th 2024–August 27th, 2024. During this time, we organized four on-site workshops and five online workshops to which we invited all member companies but also companies external to the research collaboration. This workshop series had the explicit purpose to refine and extend our previous insights and specifically, to identify the key principles of a novel product development paradigm as

well as for generating ideas for what such a paradigm could potentially look like. To allow for the emergence of new or divergent perspectives, we designed the workshops so that each workshop started with the two researchers presenting a summary of what was discussed in previous workshops. In this way, all workshop participants got an understanding for what earlier workshops contained and concluded.

The workshop series was triggered by several of the member companies asking for help and advice on how to rethink agile ways-of-working. Their concern was not the agile principles themselves, but rather the fact that the agile methods they were using did not yield the results they were expecting. Based on this input, and the fact that the companies experienced a mismatch between the theory of agile and the practice in which it is supposed to generate positive results, sparked our interest and we decided to dive deeper into this. To explore this, we decided to announce our interest and intention on LinkedIn and open up for external participation. This resulted in additional company representatives from other software-intensive systems companies but also from consulting companies and authorities in both Sweden and Denmark.

Below, we detail each on-site and each online workshop. We label them according to format: on-site workshop (OS-WS) or online workshop (OL-WS). We number them according to the sequence in which they took place (OS-WS 1–4 and OL-WS 1–5).

- OS-WS1: At the first *on-site cross-company workshop*, we had 16 people attending of which half of the group were representatives from the member companies. The workshop lasted for three hours and were documented in text, by collecting Post-it notes and by taking photos of the whiteboards that were used during the sessions. The starting-point for the discussions was that although most companies tried for the better part of two decades, they still do not see successful adoption and effective use of these practices. Especially, scaling agile practices is difficult, and regardless of frameworks like SAFe many challenges remain. Our focus of the workshop was to identify, and extend, our understanding of the key challenges with agile practices, have the companies engage in discussing these and to come up with a proposal on what an alternative paradigm could potentially look like. The explicit purpose was to move from “problem definition” to “solution development” as no practitioner nor researcher benefit from staying in problem-definition mode. As all participants were engineers and with a solution-oriented mind-set these discussions yielded interesting, provocative and highly relevant suggestions emphasizing how to approach the future of product development.
- OS-WS2: At the second *cross-company on-site workshop*, we had 40 people attending of which 20 were representatives from the member companies. Similar to the first workshop, this workshop lasted for three hours and were documented in text, by collecting Post-it notes and by taking photos of the whiteboards that were used during the sessions. Also here, the starting-point for the discussions was that although most companies tried for the better part of two decades they still do not see the expected results of using agile practices. Our focus of the workshop was again to identify and extend, our understanding of the key challenges with agile practices, have the companies engage in discussing these and to come up with a proposal on what an alternative paradigm could potentially look like.
- OS-WS3: The third on-site workshop was organized as a *company-specific workshop* where we met with representatives from case company E. The workshop lasted for four hours and was initiated by the company. At this workshop, 20 people were invited to discuss current agile practices and the challenges with these. At this workshop, our role as researchers were to facilitate the discussions, to bring our perspective on the challenges the company experienced and to together with the company representatives agree on actions to improve these challenges.

- OS-WS4: The fourth on-site workshop was organized as a *company-specific workshop* where we met with representatives from case company F. The workshop lasted for four hours and was initiated by the company. At this workshop, 5 people were invited to discuss strategy and leadership with the intention to accelerate the adoption of DevOps ways-of-working. For this company, the main challenge is the complexity of the system and how to prepare the existing architecture for future technologies. Therefore, the discussions centered around how to enable DevOps ways-of-working in a highly hierarchical organization and in systems with high coupling and dependencies between components. Both these workshops yielded results on what to improve, what to change and how to think about an alternative paradigm.

The format of the on-site workshops followed a similar structure in which the two researchers introduced by giving a presentation that explained the topic of the workshop, the motivation behind the workshop and the explicit focus on imagining a future paradigm and what this could potentially look like. The participants engaged either in group discussions or in joint discussions depending on the overall size of the group and by the end of the workshops we engaged in a joint discussion to agree and conclude on key take-aways.

We mitigated groupthink in the workshop series by structuring the process to support independent thinking, dissent and balanced participation. Before moving into plenary discussion, participants first generated ideas individually and anonymously (by using e.g., Post-it notes), which reduced early anchoring and social pressure. We then used facilitated rounds, small-group discussion (when suitable) and prioritization to ensure that all perspectives were surfaced before convergence. In all workshops, we included diverse stakeholders and cross-functional groups to reduce homogeneity of perspectives, which is a known antecedent of groupthink.

The *online workshops* were organized as open workshops to which we invited all member companies as well as companies external to the companies in the research collaboration. Similarly to the on-site workshops, the online workshops addressed topics such as e.g., Challenges with agile, Scaling of agile practices, Agile leadership, and Product management and system engineering. However, while these discussions were rewarding for refining and extending our previous insights, our explicit focus of the online workshops was to generate ideas for what a future paradigm could potentially look like.

- OL-WS1: In the first online workshop, we focused on the challenges with agile and the consequences of these. During the meeting, the participants discussed concerns with e.g., requirements engineering processes still following a waterfall process, the difficulties in aligning development and deployment, the hesitations at the customer side and the reasons for why the MVP concept does not resonate in large-scale development of complex systems. The workshop lasted for one hour and was recorded and transcribed using Microsoft Teams.
- OL-WS2: In the second online workshop, we focused on scaling of agile practices and the different approaches that the participating companies take to achieve this. During the meeting, the participants reflected on frameworks like SAFe and how such frameworks often do more harm than good as they are incredibly difficult to tailor to context-specific needs. Also, the nature of embedded systems is hard to fit into these types of frameworks which makes them challenging to apply in development of systems for which these components are critical.
- OL-WS3: In the third online workshop, we focused on agile leadership and what this implies in large organizations characterized by hierarchies and approval processes. During the meeting, the participants discussed how they had chosen to organize their agile teams, how teams reported and what was considered “good” leadership approaches. Also, some of the participants reflected on agile practices as a mechanism to increase empowerment.

- OL-WS4: In the fourth online workshop, we discussed the role of product management and how to make sure that the ‘why’ is not forgotten as agile methods typically focus on the ‘what’ and the ‘how’. The participants shared many examples of how agile has become process-heavy even if the initial ambition was to make it a light-weight method.
- OL-WS5: In the fifth online workshop, we focused our discussions on systems engineering and the many challenges that surface when adopting agile in development of systems involving physical components. The discussions revealed multiple challenges related to mismatches, governance problems, decoupling issues and engineering culture.

As the basis for analysis, notes were taken during all our interactions with the case companies. Also, we had the opportunity to record the online workshops using Microsoft Teams. During data analysis, we adopted an interpretive approach (Walsham, 1995; Maxwell, 2012) in which we used our documented impressions to carefully reflect on our learnings and what implications could be drawn from our empirical data. Similarly to the thematic analysis approach outlined in Cruzes et al. (2015) and Clarke and Braun (2017), the workshop notes were read by both researchers to identify recurring themes.

3.5. Synthesis of findings and construction of framework

Based on the two empirical components of this paper, we believe that we are able to capture the challenges that companies in the software-intensive systems domain experience in their use of agile practices. Taken together, our reflective synthesis of prior research findings and the more recent workshop series findings provides the foundation for and in-depth understanding of the challenges of applying agile practices. In the Appendix we show the mapping between the papers and workshops to the key challenges, the sub-categories that were identified and the overarching themes to which they belong.

Most importantly, the in-depth understanding of how agile practices are experienced in companies, and the challenges associated with these, provided us with a solid foundation for the construction of a framework reflecting what a future product development paradigm could potentially look like. In Section 6, and as the conceptual contribution of this paper, we present a strawman proposal called RADICAL. Our proposal is a design artifact rather than an empirical finding. In our construction of the framework, we drew on three inputs: (i) the challenges established in Section 4, which define what it must resolve; (ii) the seven principles derived in Section 5, which define the requirements the framework must satisfy; and (iii) our prior research and the related literature, from which the focus areas and the associated tools for each dimension are drawn, for example value modeling, architecture technical debt management, experimentation and data infrastructure.

In the process of developing the framework, we used the challenges we had identified (i.e., the “problems” to solve) and the principles we had derived (i.e., the “requirements/objectives” to fulfill) as drivers for our design. As mentioned above, our framework draws on three complementary inputs, each fulfilling a distinct methodological role:

First, the *challenges* identified in Section 4 serve as the problem-identification and motivation basis for the framework in that they establish the specific issues the framework is intended to resolve. Thus, the challenges anchor the framework’s relevance to a concrete empirical problem space.

Second, the *principles* derived in Section 5 operationalize these challenges into a set of design objectives. Rather than resolving the challenges directly, the principles specify the requirements that any potential solution — including the proposed framework — must satisfy to adequately address the identified challenges. Our procedure followed thematic synthesis as described by Cruzes et al. (2015) for synthesizing findings across case studies, and thematic analysis as described by

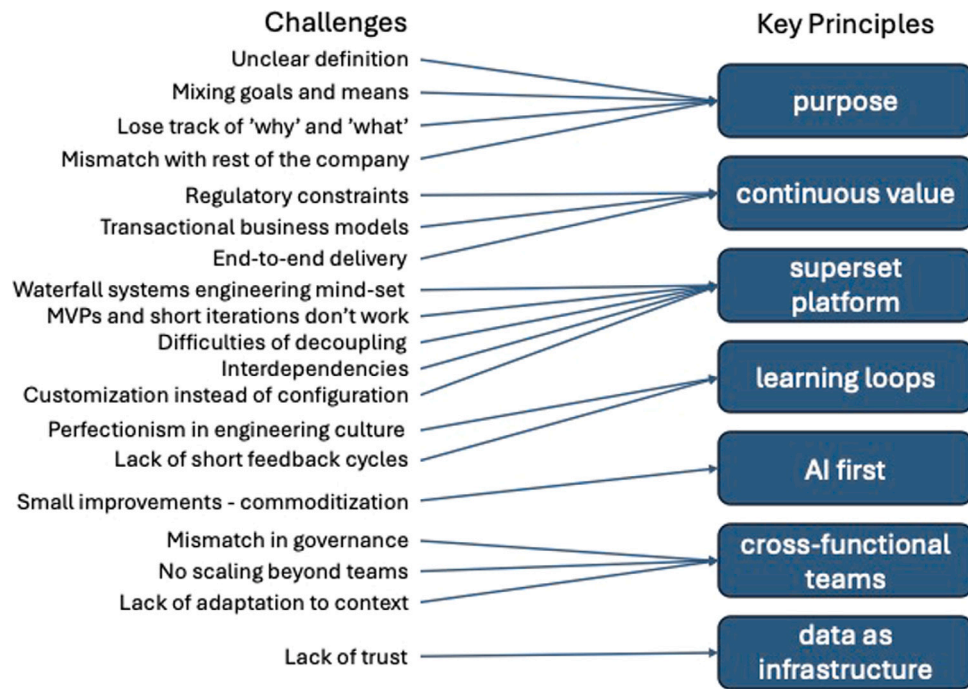


Fig. 2. Relating key challenges of agile to key principles of a new paradigm.

Clarke and Braun (2017), in which themes are understood to be actively constructed by the researcher through engagement with the empirical data. The mapping was produced as follows: the challenges established in Section 4 were compared pairwise for shared underlying concerns; challenges expressing a common concern were clustered; each cluster was then labeled with the principle that addresses that concern; and the resulting clusters were reviewed against the workshop material to check that the labels corresponded to concerns that participants had themselves articulated. For example, ‘unclear definition’, ‘mixing goals and means’, ‘lose track of ‘why and what’ and ‘mismatch with rest of the company’ were all experienced as challenges that had to do with ‘purpose’. Several workshop participants refined, and added to, the challenges and shared their view on a certain challenge or a perspective that were missing e.g., perspectives on definitions of agile and how people often mix goals and means. As another example, people shared how agile teams “lose sight” of the overall picture (i.e., the ‘what’ and the ‘why’) and find it difficult to align with the rest of the company. In our analysis of this input, we group these experiences together and labeled it ‘purpose’. To address these challenges, the ‘purpose’ is then one of the key principles needed in a new paradigm. We did the similar exercise for all challenges and the result of our analysis is presented in Fig. 2. It should be noted that several of the challenges we identified could relate meaningfully to more than one principle. Therefore, what Fig. 2 depicts is, for each challenge, the principle we judged to be the primary one addressing it on the basis of how practitioners framed the challenge in our material, it does not assert that no other principle could be relevant.

Third, our prior research and, to some extent, related literature provide the theoretical and empirical grounding from which the framework’s constituent elements are derived. This knowledge base was used to identify (i) the focus areas that constitute each dimension of the framework, and (ii) the tools, practices, or techniques associated with each focus area. This step ensures that the framework’s components are not proposed ad hoc, but are drawn from, and consistent with, our accumulated knowledge about the companies and what they experience as well as established work in the field that describes similar situations.

Together, these three inputs (challenges - principles - prior work) form a traceable derivation chain: challenges define what must be resolved, principles define what a solution must satisfy to resolve it, and the literature-derived focus areas and tools define how each requirement is concretely addressed within the framework.

4. Empirical findings

The case companies involved in our research build systems including mechanics and electronics and for which software, data and artificial intelligence (AI) are increasingly important (Kasauli et al., 2021, 2018; Habibullah et al., 2024b,a; Eklund and Berger, 2017). All case companies work in an agile fashion with the intent to realize short iterations, fast feedback loops and increased responsiveness to customers. The majority of the case companies have adopted DevOps practices and several case companies use the SAFe framework.

In Section 4.1, we present the key challenges that the companies experience and that they, over time, have raised as their main concerns with agile practices. The identification of the key challenges constitutes the response to RQ1. The key challenges we present are our generalization of the insights we accumulated during 14 years of close collaboration with the companies. In the more recent workshops, we confirmed these insights while also collecting new additional insights. Hence, the challenges are empirically derived from all these activities and they represent challenges that the majority, or all, companies experience. While there might be minor variations depending on company and industry domain, the overall experiences are shared among the companies. In Section 4.2, we reflect on why these challenges occur and what is missing in existing agile methods.

4.1. Key challenges

Below, we detail the key challenges that the case companies involved in our research experience with regards to agile practices and scaling of these. As a structure for this, we group the challenges in the following four categories: (1) *Definitions and interpretations*, (2) *Business and monetization*, (3) *Systems engineering*, and (4) *Organizational culture and ways-of-working*.

4.1.1. Definitions and interpretations of agile

Unclear definition: In the case companies we studied, there is no agreement on what agile means, what agile practices are and for what purposes these should be used. In the companies, different teams and different functions have their own definition of agile, and within each company there are as many definitions as there are people (if not more). This results in a situation in which people typically assume that they agree on what agile is and how to best work according to its practices. However, when discussing concrete practices and what these mean in practice, there is no agreement and people do not recognize each others' interpretations. At times, this challenge causes a situation in which teams rather hold on to the waterfall mind-set and development approach as this is easier than trying to align and agree on what agile is or should be.

Regulatory constraints: The case companies in our research tend to build systems including mechanics and electronics in addition to software and machine learning models. As these systems tend to be used in safety-critical contexts, there are several regulations that apply. Consequently, each product or product release needs to be certified to ensure that the product meets the regulations. Depending on the industry, certification is done by external certification bodies or companies can self-certify, but there is often a significant cost associated with certification. Part of the challenge is that regulatory authorities tend to assume a waterfall development and validation approach and hence the certification tends to be a significant undertaking. Although several proposals for iterative certification exist, e.g. SafeScrum (Hanssen et al., 2018), these approaches have not yet reached the maturity stage needed for deployment. This makes the adoption of DevOps difficult if not impossible in several of the case companies we studied.

Lack of adaptation to context: In many companies, agile practices are perceived as generally applicable and useful across the organization, across products and across different types of technologies. While this might be true in theory, practice shows that agile practices need to be carefully adjusted and tailored to fit each individual organization and its needs. During our research, we heard company representatives talk about agile as "one size fits all" and recognizing this as a major problem and one reason for why the adoption of these practices are so hard in software-intensive embedded systems companies involving fundamentally different technologies and competencies.

Mixing goals and means: Often, agile is perceived as a goal in and by itself rather than a means to an end. In practice, this means that companies focus their efforts on adopting the agile processes and tools in the optimal way rather than focusing on the outcomes they seek to achieve. In the companies we studied, endless efforts are spent on processes, roles and ways-of-working and every time there is a new version of a method and framework (e.g., SAFe) there is a new training program being kicked off. While people appreciate this in the sense that they learn new things, they also report on a situation in which the new methods and frameworks they need to learn about brings focus away from the product itself.

4.1.2. Business and monetization

Lose track of 'why' and 'what': Agile practices focus on the 'how' rather than on the 'why' and the 'what' of development. As a result, organizations become process-driven and lose sight of what they aim to accomplish and why this is important (Olsson and Bosch, 2025). Typically, more than half of all the features in a typical software system are never used (Backlund et al., 2014). In our experience, this is due to a too strong feature focus and a too vague idea of what adds value to customers. Ironically, agile was intended to help teams move away from process-driven ways-of-working, but with its heavy emphasis on 'how', agile is by many perceived as an as heavy process as many of its ancestors.

Transactional business models: Traditionally, the companies in our study have employed a transactional business model where the company gets paid for the product it sells to the customer at the time of

sale. As these companies tend to align their revenue and cost models, there often is resistance towards DevOps as this would imply continuous cost for creating each release, but there is no recurring revenue. Most of our case companies are experimenting with continuous business models to complement their transactional revenue, but far from all experiments are successful. As the customer has already paid for the product, it is hard to convince to customer to part with more money to get additional functionality. Instead, many customers feel that they are entitled to this functionality as they already bought the product.

MVPs and short iterations do not work: In most cases, the systems that the case companies develop need to be fully functioning before they can be tested as required. This means that having e.g., a minimal viable product (MVP) with only parts of the functionality in place is not an option. Rather than enabling testing with customers, as is the intention with an MVP, it can actually cause e.g., vehicles to not be allowed to operate on public roads. Typically, testing with customers requires about 80% of the new functionality in place. In all case companies, being able to run realistic tests and/or experiments requires the companies to build most of the functionality that is intended to be part of the product. Otherwise, the perceived or actual business benefits for the customer are not realized. The result is that the role of experimentation is "non-existing" (or very small).

4.1.3. Systems engineering

Waterfall systems engineering mind-set: Despite the amount of software in an embedded system, a large portion of the organizations we studied are still reasoning in terms of hardware components and waterfall ways-of-working. For example, in the automotive case companies, people tend to think about the interaction between electronic control units (ECUs) rather than the interacting software elements. While software is increasingly important, the mind-set is still very much "mechanics". This means that people in the case companies view things very differently depending on whether they put the software functionality in the forefront, or whether they take the hardware perspective on the system. In the case companies, the hardware mind-set is still very strong since often, the mechanic part of the organization used to outsource electronics and software to suppliers rather than developing these components in-house. This means that the agile mind-set, advocating incremental and iterative development, is very difficult to communicate, adopt and benefit from.

Perfectionism in engineering culture: In theory, most people appreciate the idea of iterative development and continuous improvement of functionality. In practice, most people in the companies we studied tend to adopt a "We need to be perfect before we ship" mind-set for which agile principles were never designed. This engineering culture makes anything except waterfall development of the full product very difficult and people find it very hard to understand why customers would appreciate receiving functionality that is not complete and fully functional.

End-to-end delivery: Reaching business agility and the continuous delivery of value to customers through, among others, DevOps requires many functions in the company to be involved. From a systems engineering perspective, a new version of the software needs to be tested together with the mechanical and electronic components in order to ensure that no unintended behavior surfaces. This requires the departments responsible for mechanics and electronics as well as for system quality to allocate resources continuously in order to support this, but in the case companies these functions are organized around waterfall, one-off processes and getting the resources is difficult, if not impossible.

Difficulties in decoupling: Feature teams and decoupling of teams does not work in development of large and complex systems. In a large embedded system, there are many things that needs to work together to achieve a fully functioning system. Hence, different streams, trains and/or teams cannot be decoupled and work in isolation from each other. Instead, they need to collaboratively resolve issues and

they need to understand each other. In the case companies, there is a lot of synchronization that is needed between teams to get all the functionality, all the components and all the elements working together as a system.

Lack of trust: Architecture and decoupling of teams is a mechanism to enable different teams to make progress within their area of expertise. As long as all teams do what they are supposed to do this model works and there is a product. However, in practice there is typically not the trust that is needed in between different teams to deliver their parts. As a result, there is no end-to-end solution.

4.1.4. Organizational culture and ways-of-working

Mismatch with rest of the company: The challenge of continuous rather than waterfall-style processes not only applies to systems engineering, as discussed in the previous section, but also to other functions such as customer support, sales and product maintenance. Everyone needs to be aware of new software releases and the new value as well as issues that might surface as part the release. This requires these functions to allocate resources continuously in order to support this, but in the case companies these functions are organized around waterfall, one-off processes and getting the resources is virtually impossible. As an example, one of the companies that managed to get to bi-weekly releases of software to its customers ran into the problem that customer support was always behind in their understanding of the issues that were surfacing at the customer. By the time the customer support teams were getting on top of a release, the next release had already been rolled out and all obtained knowledge was no longer relevant.

Mismatch in governance: Many of the case companies coordinate across functions and departments using a stage-gate or milestone model. This model is not only used for alignment, but also for resource allocation and go/no-go decisions. Projects can be stopped at each stage-gate if the results are not satisfactory. Although important as a tool to avoid wasting resources, it is entirely at odds with the “perpetual beta” culture in agile. In the traditional model, teams are asked to make promises about functionality to be delivered at a certain date in the future, which often collides with the sprint planning and fast cycles model in agile.

Interdependencies: In the case companies we studied, there is not a single unit that can deliver functionality completely by itself. Instead, the systems that the case companies develop are highly interconnected with dependencies and relationships between multiple components and technologies. Hence, cross-team and cross-organization collaboration is required at all stages but is difficult as response time in between different teams, functions, units and organizations is typically slow. In addition, cycle times between different teams, functions, units and organizations vary.

Customization instead of configuration: To shorten cycles and reap the benefits of agile practices in software-intensive embedded systems, traditional customization of products needs to become configuration of products. In practice, this means that there needs to be a shift from a subset platform to a super-set platform where every delivery to a customer is derived from the super-set platform and configured for the specific customer. Without this model, it is often prohibitively expensive to deploy and deliver frequently to customers and to adopt DevOps ways-of-working.

Lack of short feedback cycles: Agile development assumes short feedback cycles which is not the reality for B2B companies. The case companies build highly complex products with very limited opportunities to validate with customers during development. Typically, there is no direct access to the end-user as the companies operate in a B2B context. The lack of feedback loops, or very slow feedback loops, make practices such as iterations and continuous improvements difficult to implement and benefit from.

Small improvements - commoditization: In the case companies, the use of agile methods tends to establish a culture in which “the more features the better”, i.e., teams focus on delivering as many features as

they possibly can in a certain time frame. What easily is forgotten in a culture like this is what customers really appreciate. Throughout our research, we experienced situations in which companies use agile methods but without distinguishing between commodity, differentiating and innovative functionality. As a consequence, many companies face a situation in which efforts that go into commodity are too high while the resources for developing differentiating and innovative functionality are limited. Over time, this drains the development organization and there is no room for innovation and for optimizing functionality that brings new revenue.

No scaling beyond teams: While all case companies have adopted agile practices at a team-level in their software development organizations, any attempt to scale beyond this has failed. To the best of our knowledge, only one of the case companies has succeeded in scaling agile practices beyond the software teams. Still, these deployments are only used for testing purposes and hence, are not deployments that benefit the customer on a frequent basis.

4.2. Why do these challenges occur?

One might wonder why a paradigm as popular and widely established as the agile paradigm would result in the challenges that we identify and discuss in the previous section. In [Iivari and Iivari \(2011\)](#), the authors recognize how the adoption of agile methods is a trend but not necessarily unproblematic. As one of the reasons for this, the authors discuss compatibilities and incompatibilities between agile methods and organizational culture. In the context of agile adoption, the richness of the concept of “organizational culture”, the ambiguity of the concept of agility, and the variety of ways the two may be related with each other result in problems ([Iivari and Iivari, 2011](#)). Often, agility is perceived as a cultural change more than anything else and this makes measuring the agile maturity of teams and organizations difficult ([Gren et al., 2015](#)). Throughout the years, many researchers have studied the challenges associated with agile practices and we are not the only ones recognizing the slow and difficult adoption process. In [Paasivaara et al. \(2018\)](#), the authors emphasize how large-scale agile transformations are seldom easy, and that literature provides little advice on how to successfully proceed. In particular, the authors recognize that current agile approaches do not provide good blueprints for what a scaled agile organization should look like, and that the recent scaling frameworks are largely unvalidated. In addition, the authors recognize how the lack of a clear and common understanding of agile led to a lack of a unified agile mindset in the company they studied. Based on this, they conclude that giving teams autonomy without enough coaching leads to a suboptimal implementation of agile. In a rigorous study by [Dikert et al. \(2016\)](#), the authors identify 35 challenges of which many are related to the implementation process being difficult due to the lack of a clear and common definition of agile, poor customization opportunities and difficulties of applying it to a large-scale setting. Although there are several other studies on the similar topic, we believe it is fair to say that the challenges of agile are well recognized and that many studies report on cases where organizations struggle to adopt agile ways of working.

In our experience, the heart of the challenge is at the origin of the agile manifesto. The manifesto was defined by a group of consultants that worked as a single team with a specific customer that they had easy and continuous access to and for whom they built bespoke software. Although there are many developers that work in that context, the software-intensive embedded systems companies involved in our research experience a very different reality. These companies have many customers, many teams and many products that often are connected via common mechanical, electrical and software components.

The agile paradigm, in part, received such a strong and positive response as it addressed a frustration that many had with the then prevalent Capability Maturity Model (CMMI) e.g., [Royce \(2002\)](#), [Jones and](#)

Soule (2002). Although CMMI never was intended in this way, its realization in many companies was very process-heavy resulting in many engineers feeling like they were operating in a straitjacket without any opportunity for innovation and a severe lack of agency. Although some complained about the “agile cowboys” in the early years, in many contexts, especially pure software systems, agile teams had significantly better outcomes than traditional process-heavy approaches.

However, due to the roots of the agile paradigm, many aspects of modern product development are not required and consequently not part of the paradigm. At the root of the challenge, we see a lack of scalability. Despite prominent research on business agility (Mathiassen and Pries-Heje, 2006; Setiawati et al., 2022; Olsson and Bosch, 2024b), agile maturity (Yürüm et al., 2018; Turetken et al., 2017) and agile culture, Iivari and Iivari (2011), as well as practical implementations such as Scrum of Scrums and SAgE, the scalability issue remains and the companies we studied keep struggling with how to extend the agile principles beyond their software teams and organizations. Specifically, *product management, systems engineering and business models* are mechanisms that companies traditionally employed to scale their business from a project-centric consultancy business to a product, product portfolio or platform business.

Product management is concerned with collecting, integrating, harmonizing and prioritizing requests from customers as well as roadmap functionality based on internal analysis of trends and predictions of future customer needs. Especially in the companies involved in our research, products include mechanical and electronic components in addition to the software and machine learning components and the former need significantly longer lead times to make available to customers.

Systems engineering seeks to deliver end-to-end system functionality to customers that meets the specific use cases that these companies use the product for. Although many systems engineers and architects understand the value of realizing functionality in “bits”, i.e. through software and machine learning models, the mechanical and electronic components are an integral part of the system. These components can only be developed iteratively in sprints during the design phase, often making extensive use of simulation. However, once we reach the manufacturing and deployment of these components, iterative development becomes prohibitively expensive. Consequently, software-intensive systems companies need to develop mechanisms to align the “atoms” and the “bits”.

Finally, the goal of any company is to generate revenue and positive margins on that revenue. This means that the company needs a *business model* that allows it to capture part of the value it delivers to customers as a financial reimbursement. Typically, software-intensive systems companies seek to align their revenue and cost models. Because of that, companies that use a transactional business model often resist DevOps practices as this would imply continuous cost for which there is no matching recurring revenue. Of course, these companies seek to employ continuous business models to complement their transactional revenue to address this problem, but customers are often hesitant to commit to continuous cost unless the value proposition is sufficiently compelling.

Of course, we are far from the first to identify this and most agile frameworks seek to address one or more dimensions of the scalability problem, including Scrum (Schwaber, 1997), SAgE² and others. The problem is that these frameworks heavily borrow from the traditional waterfall-style approaches and, as a consequence, result in implementations at companies that tend to be akin to the Emperor’s new clothes. In the words of one of the interviewees at one of the companies involved in our research: “We adopted SAgE, but nothing changed”.

In our experience, most practitioners and researchers fail to recognize that agile is a means to an end, rather than a goal in itself. In the end, the goal is *business agility*, i.e. the ability for a company to rapidly respond to requests from customers, market developments and

societal and technological trends. In Mathiassen and Pries-Heje (2006), the concept of business agility is understood as organizations that are capable of implementing change, that are nimble, and that are capable of responding quickly and gracefully to both expected and unexpected events in their environment. In our experience, a continuous business model is critical for achieving this as it allows companies to not only deliver value on a frequent basis but also capture parts of that value (Olsson and Bosch, 2024b). Hence, for software-intensive system companies, agility in software is only one aspect that is necessary but far from sufficient. Based on the insights from working with the companies involved in this research for more than a decade, we see the need for an alternative approach, i.e., a new development paradigm. To address the challenges we identified, and to help companies solve the mismatch they experience between the agile practices and the reality of product development, we have strong reasons to believe that there is a need for a novel product development paradigm and that staying with the same approach will not lead to the business agility that companies are striving for.

5. Key principles for a new paradigm

A recurring theme in our research is that the challenges of agile cannot be solved with the same thinking that produced them; addressing them requires a different starting point. This is also true for the thinking surrounding the agile paradigm and our conclusion based on the key challenges that we presented in the previous section and the numerous interactions with the companies that we work with is that we need to develop a new product development paradigm. Below, and in response to RQ2, we present seven principles that a new development paradigm should follow, i.e., the requirements such a framework must satisfy. In Fig. 2, we present these seven principles and we map the relationship to the key challenges that we identified in Section 4.1.

Purpose: The first principle is concerned with purpose. With millennials and Gen-Zs entering the workforce, several of the companies involved in our research expressed the importance of their company stating and explicitly living up to a clear purpose. This makes good economic sense and research dating back all the way to 1992. Kotter and Heskett (1992) shows that companies with a clear purpose grow revenue, on average, 5 times faster and have 10 times the stock price of peers that focus on revenue and profit only. All workshop participants agreed on that purpose is critical and that this is something that has gotten lost in process-heavy agile practices where the focus is on the next feature rather than on the overall vision and goal of the product and the company.

For a company to be clear on its purpose addresses several of the identified challenges. It helps with the unclear definition of agile as it becomes a means to an end, i.e. delivering on the purpose rather than following the agile process as a goal in itself. The same argument applies to the challenge of mixing goals and means as well companies losing track of ‘why’ and ‘what’. Finally, in companies where software R&D operates in a way that does not match the ways of operating in the rest of the company, the purpose helps set a north star that everyone in the company can align on.

Continuous value: The second principle is concerned with continuous value delivery to customers as well as the continuous capture of value from customers. Many of the companies that we worked with are employing DevOps, meaning that customers receive new functionality and features on a continuous basis. However, most companies have difficulty adopting continuous business models to generate revenue from the improvements that they provide to their customers. As can be seen in Fig. 2, there are several challenges that bring us to this principle. In the workshop series, people shared how they operate under increasing regulatory constraints, how difficult it is to shift from traditional business models and how end-to-end delivery of features is still challenging. All of these challenges are evident in the agile

² <https://scaledagileframework.com/>

paradigm and despite various efforts and attempts teams and organizations cannot solve them. There is simply very little focus and support for these in the current development paradigm.

The principle of continuous value addresses several of the identified challenges. With such a principle in place, there is continuous value capture, there is a revenue incentive to address regulatory constraints and it forces companies to explore, experiment with and adopt new business models that allow them to capture part of the value that they deliver to customers. Finally, it helps companies align their entire value delivery chain, and not just the software part, to the continuous nature of value delivery. It should be noted that this principle is high priority in most companies and the notion of continuous value delivery and capture was mentioned frequently, across companies and as one of the most important factors as it could help address many of today's challenges.

Superset platform: Another key learning from especially the workshop series is that companies need to quite fundamentally rethink their product architectures, reuse of existing code and variability in the product portfolio when adopting DevOps as a means to achieve continuous value delivery. Rather than building software for a product once, it now requires new software releases every two to four weeks which induces major overhead if each product is managed by itself. Instead, companies are increasingly pursuing the notion of a superset platform. This platform contains the superset of all functionality in all products. The consequence is that each product basically becomes a configuration of the platform and there is no product specific functionality left. This also means that extensions to any of the components or subsystems in the platform become available to all product configurations and the decision to offer the extension for a particular product becomes a business decision rather than a technical one.

The superset platform principle addresses several of the challenges that we identified in current agile practices. As the superset platform not only includes software, but also mechanical and electronics components, the waterfall mindset often found in systems engineering is replaced with a continuous or iterative mindset where mechanics, electronics and software all evolve continuously, but at different speeds. Also, it addresses many of the challenges around MVPs as any MVP can be derived as a configuration of the common platform and configured with the additional functionality that innovation teams might want to validate. Third, decoupling and inter-dependencies are explicitly managed as part of the superset platform architecture and stop being late surprises in product development or customer projects. Finally, the superset platform principle removes the need for customization and replaces it with configuration. In our view, this is a critical principle for a new paradigm as it allows teams and organizations to be responsive, to increase value delivery to customers and to evolve the product at a pace that is not feasible in current agile ways-of-working.

Learning loops: A key difference in our thinking that appeared during this research is that traditionally companies focused on building functions and features that either were realized or not. With the adoption of DevOps, the focus is shifting towards capabilities that have quantitative KPIs associated with them. This allows companies to use the iterative deployment to experiment with the realization of functions and features and to measure the impact on these KPIs. The goal for teams then becomes to focus on improving capabilities rather than building endless lists of features that customers do not use, which is often the case in today's agile development paradigm, also referred to as "featuritis" (Bianchi et al., 2018; Shmueli and Ronen, 2017).

To focus on learning loops addresses several of the challenges that practitioners experience. First, the focus on building the perfect feature or function is significantly diminished as we continuously iterate to improve a capability rather than "the feature". Second, learning loops work the best when there is frequent feedback and hence the company tends to organize around feedback loops to feed the learning loops with the needed data. With this principle in place, there are incentives to

evolve and to improve a capability and teams can adopt a long-term rather than a short-term "featuritis" focus.

AI first: The key technology that has emerged over the last decade and that was not available when agile was initially defined is, of course, artificial intelligence (AI). The use of machine learning, deep learning, reinforcement learning, federated learning and Agentic AI is widely experimented with in the companies involved in this study and several deployments are already in place. As a result, the companies are increasingly concerned with organizing their business processes, R&D processes and product functionality from an AI first perspective. This means that everything you do is first designed to maximize use of, and benefit from, AI functionality and only where this does not lead to satisfactory outcomes are humans brought into the process e.g., for monitoring, human-in-the-loop, validation of results and/or recommendations.

While the AI first principle could actually address many of the challenges we identified, the most important one (as recognized by the workshop participants) is that it helps solve the focus on small improvements which is typically the case in today's agile organizations. Teams work on iterative development and incremental improvements without allowing themselves or without recognizing the need for major improvements and opportunities to do things in a completely different way. When applying the AI first principle, the focus is on major improvements in functionality in response to the powerful capabilities that can be realized using AI.

Cross-functional teams: Although the term 'cross-functional teams' is used in agile practices and throughout the agile history, it is often used to refer to teams that include the various R&D functions, such as development, architecture and quality assurance. However, several of the companies we work with are now exploring "true" cross-functional teams that can deliver functionality, features and improvements end-to-end. In this case, these teams are associated with capabilities and asked to improve these capabilities (instead of working with a single feature in focus). To avoid local optimization, such teams have guardrail metrics that they need to respect and no improvements are accepted if these violate the guardrails. In this way, cross-functional teams operate end-to-end rather than within a feature scope.

Organizing teams as true cross-functional teams addresses several challenges of agile practices. First, the governance issue is resolved as teams are assigned to capabilities that have been prioritized by central product management or "the business". In that sense, the model allows for centralized decision-making and decentralized execution. Second, the principle of "true" cross-functional teams allows teams to fully and independently develop and deploy their functionality and hence, the approach scales to the entire organization (and this is feasible without adopting complex frameworks such as SAFe that was intended for scaling agile but that the companies we work with experience as process-heavy, difficult to adapt to their needs and expensive to maintain). Finally, the focus on capabilities and quantitative KPIs allows the organization to adjust its ways-of-working to the specific context in which it operates.

Data as infrastructure: Data has played a significant role for decades in the companies involved in this study. However, much of that data is collected, stored and used in silos and optimized for specific use cases (and this is often enforced in agile organizations where teams collect and use data for a specific feature and for their own use). As the semantics of the data and its quality often is questionable, the result is that many companies do not trust the data. Especially in situations where opinions of senior leaders and data are at odds with each other, the data is often reinterpreted or ignored. By ensuring the quality, semantics and accessibility of data, i.e. building a proper infrastructure around it, companies can rely on it and enforce true data-driven practices.

Establishing an infrastructure around data allows for trust in the organization, which is one of the key challenges that we identified. Also, it allows for teams and senior leaders to measure and establish

progress and value based on data that is collected, shared and used is a structured, strategic and systematic fashion across the company and its different functions.

Concluding, the seven principles that we described above form the basis for a new perspective of product development that is not an extension or evolution of the agile paradigm, but rather a new paradigm that we believe is emerging in the software-intensive industry. While many of the principles and practices underlying agile will be present in this new paradigm, they will be used in a different context (the era of AI), evaluated from an alternative viewpoint (continuous value delivery) and driven from a perspective that is quite different from traditional product development. All company representatives involved in this study, as well as our previous findings, show that there are indeed benefits with agile practices. However, since the introduction of agile practices the ways in which the software-intensive industry operates have completely changed. To a large extent, agile optimizes how teams work. In today's product development, organizations need support for how to learn, decide and create economic value continuously and at scale.

6. RADICAL: A strawman proposal

In this section, we present a conceptual framework that is our strawman proposal of what a new product development paradigm could potentially look like. Our proposal is a conceptual contribution and a design artifact informed by the empirical results and by prior literature rather than a finding that follows from the data alone. The conceptual framework is our response to RQ3. In our construction of the framework, we drew on three inputs: (i) the challenges established in Section 4, which define what it must resolve; (ii) the seven principles derived in Section 5, which define the requirements the framework must satisfy; and (iii) our prior research and related literature, from which the focus areas and the associated tools for each dimension are drawn.

The working name of the proposal is an acronym as well as a word: RADICAL. RADICAL is intended as a holistic product development paradigm encompassing business, product management, systems engineering, software engineering, quality assurance, release organization, customer support, market units, data collection and processing, etc. in one paradigm. Our intention with the proposal is to help address the many challenges with agile as discussed in the earlier section in this paper as well as in previous literature, see e.g., [Paasivaara et al. \(2018\)](#), [Dikert et al. \(2016\)](#), [Kurapati et al. \(2012\)](#), [Eklund and Berger \(2017\)](#), [Laanti et al. \(2011\)](#).

In Section 6.1, we discuss the basis for the RADICAL framework and why it, in our view, constitutes a new paradigm for product development and evolution. In Section 6.2, we provide a high-level overview of the concepts in order to provide a sketch of a future development paradigm to replace agile. Finally, in Section 6.3, we provide illustrative examples of how some of the companies are employing part of RADICAL.

6.1. Why RADICAL is a new paradigm

In our view, RADICAL is a new paradigm that is not an extension of agile, but rather a different way of approaching product development and evolution. It uses several of the agile principles, but it is different in three ways; (1) *company versus software R&D*, (2) *systems versus software* and (3) *infinite vs finite games*. Below we discuss each of these in more detail.

Even though there are some attempts to apply agile to electronics design, fundamentally agile is a process-driven approach to software development. Even though it was introduced to replace the process heavy CMMi, also agile defines a clear process consisting of sprints, retrospectives, sprint planning and other rituals. In addition, and according to most practitioners in the companies we work with, agile

pays little attention to what gets built and whether the developed functionality adds value. Instead, it focuses primarily on how to build the software and on delivering what gets requested by customers, product management or sales. Although agile practices come with good intentions which is the reason for why companies seek to adopt it, the adoption and use of these practices remain very challenging in large-scale systems development. Finally, agile uses a backlog-driven development approach and does not measure the impact of developed functionality in terms of the value delivered to customers. In our view, agile is focused on the efficiency of software R&D.

RADICAL, on the other hand, is value driven and uses quantitative value modeling as a basis for development of new functionality. Similar to 'Objectives and Results' schemes (OKRs), it is effective for setting goals and aligning on these, for communicating what you want to accomplish and what milestones you will need to meet to accomplish the goal. As reported by [Stray et al. \(2022\)](#), OKRs provide benefits such as focus and frequent priority setting and it allows teams to cross-functionally align and create better products. However, while the OKR approach was designed to help organizations achieve their business goals in a more efficient way through employee engagement, the value modeling approach we advocate has its emphasis on quantitative metrics and development *effectiveness*, i.e. the amount of business value created for every unit of R&D.

RADICAL explicitly focuses on minimizing investment in commodity functionality by labeling all functionality in the system as being commodity, differentiating or innovative and prioritizing the allocation of R&D resources accordingly. Third, it treats requirements as hypotheses that are to be tested through experimentation to determine whether new functionality delivers value to customers. This means that RADICAL includes product management and is closely connected to other company functions, such as sales, to achieve R&D effectiveness.

Agile systems engineering approaches such as [Dove \(2005\)](#), [Turner \(2007\)](#) and [Douglass \(2015\)](#), start from the requirements, rather than measureable outcomes, and focus on the design phase of systems. The post-deployment phase where new value could be added not only through software deployments, but also through updated electronic and software components, is largely ignored.

Second, whereas agile is predominantly focused on software, RADICAL is a comprehensive product development paradigm that includes mechanics and electronics into its scope. That means that RADICAL also includes systems engineering. In [Fig. 3](#), we illustrate the key concepts underlying Radical. First, it assumes that all technologies are subject to continuous evolution, not only software. The main difference is that the frequency at which the different technologies evolve is different due to the economics of R&D, manufacturing and deployment for the different technologies.

RADICAL takes a (superset) platform centric approach where each product is a configuration of the functionality in the platform, including mechanical, electronic, software and AI components. All the latest versions as well as the variants of components are part of this superset system platform and all products in the product portfolio are merely configurations of the superset platform. This also means new products, existing products during manufacturing as well as products deployed in the field are in principle eligible for the new versions of these components. Whether deployed products get these updated components depends on the business model and realized value. This requires that the company is able to monetize the improvements that it provides.

Third, both traditional and agile systems engineering approaches as well as agile are inherently project focused, meaning that a product is developed in a project with a clear start and end. In that sense, these are concerned with "playing a finite game" in the definition of [Carse \(2011\)](#). Traditional systems engineering as well as agile projects are concerned with delivering a product to the customer and getting paid for it. Although DevOps has become widely adopted in SaaS companies, the software-intensive embedded systems industry is still in the transition towards it.

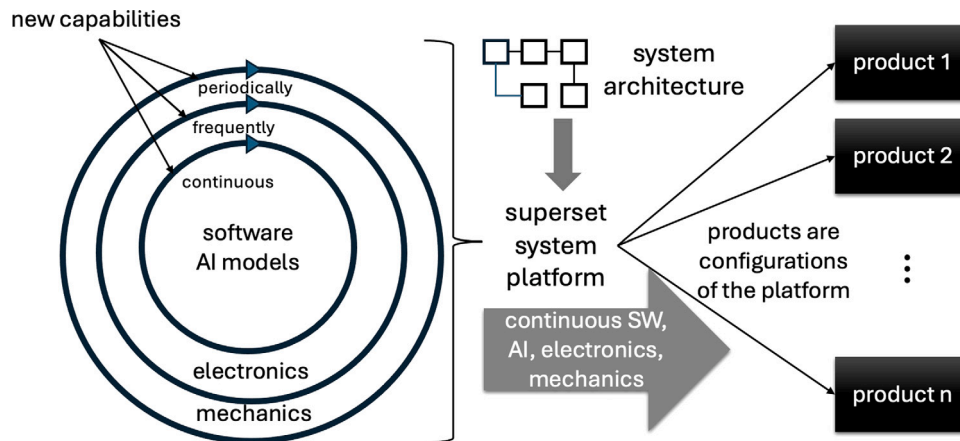


Fig. 3. RADICAL perspective on systems engineering.

RADICAL, on the other hand, is concerned with playing an infinite game. The company and its customers have a continuous relationship where the company is providing continuously improving value to its customers and is being rewarded for it. There is a constant feedback loop between customers and company through quantitative and qualitative data as such the company can track the value it delivers to customers, measure the impact of experiments and change the value model as the preferences of customers change.

6.2. RADICAL framework

As we mentioned in the introduction of this section, RADICAL is both a word and an acronym. In the text below, we discuss each of the letters of RADICAL by providing a definition, discuss focus areas as well as the tools that can be used to deal with this aspect of the framework.

6.2.1. R: Responsive

Definition: In essence, agile is a reactive paradigm and much of the conversation is about responding to change. The basic notion is that we exist in some kind of steady state, there is an external disruption that requires us to react and we use agility to return to the steady state. Being responsive is different in that there is a strong sense of direction and intended destination. When there are external forces, we respond by incorporating these forces, without losing the direction we have set.

Focus areas: Three aspects of responsiveness deserve to be discussed: vision and definition of success, triaging external forces and redefining success. In short, we need to know where we are going and how we plan to get there. Then, when disruptions or triggers occur, we need to know which ones need to be responded to and which ones can be ignored. Finally, there are times when the initial destination is no longer the best one and we need to redefine it.

Tools: Vision and business strategy are typically outside of the scope of software engineering and we refer to the business literature. However, we have developed techniques such as value modeling (Olsson and Bosch, 2018a, 2020a) to express the business strategy in quantitative terms. This allows for real-time tracking of the implementation of the business strategy.

6.2.2. A: Architecture

Definition: As agile was in part a response to CMM and process-heavy approaches with significant focus on requirements and architecture rather than code, the two have always been at odds. Also, the agile manifesto prioritizes human interaction. The challenge is that coordination through meetings, which is what human interaction often is about, is orders of magnitude less efficient than coordination through

architectural interfaces. In many of the companies we studied, meetings are often viewed as very effort consuming and error prone as it is easy for different team members to misinterpret the agreements from the meeting, causing further cost and inefficiencies. Hence, to have two teams work on different sides of an API is more efficient as long as the API provides the functionality required by both teams. As long as teams stay on their side of the API, there is no need to interact. Although this requires careful API design, in the end this is what software architects are concerned with.

Focus areas: Architecture in many ways is a highly overloaded term, but in RADICAL we believe organizations need to focus on three aspects: the relation between business strategy and architecture, system and software architecture and architecture refactoring. Many engineers fail to recognize the, sometimes enormous, implications of architecture design decisions on business strategy options. Second, especially for systems including mechanical and electronic components or where an externally enforced architecture constrains architecture choices, we need to clearly define interfaces at multiple levels, including technical, ways of working and organization. Finally, like the rest of the system, architectures require continuous evolution and this evolution needs to be managed through architecture refactoring.

Tools: In earlier research, we have worked extensively on architecture technical debt management (Martini and Bosch, 2019). These techniques allow for the identification of technical debt, which typically is the mismatch between business strategy and architecture, the prioritization of this debt based on business impact and the realization of the refactoring.

6.2.3. D: Data-driven

Definition: For all the terabytes and petabytes that they have stored, most decisions in the companies that we work with are still opinion-based. Where conflicts with the data exist, the data is reinterpreted to match opinions. We need to invert this way of operating and ensure that we start from the data and proceed from there to decisions where we allow qualitative data and 'bets' to enter the process.

Focus areas: We can take several perspectives concerning data, but based on our findings, the three main aspects should include: defining outcomes quantitatively, using experimentation to manage uncertainty and data management. We need to define the intended outcomes of actions in the company in quantitative terms to make sure that we are making real progress. Second, uncertainty is a fact of life in virtually anything we do. We can use experimentation to decrease uncertainty with limited use of resources. Finally, the infrastructure we use for collecting, processing and storing data as well as managing data assets requires careful design by the company.

Tools: We mentioned value modeling (Olsson and Bosch, 2018a, 2020a) earlier, but data-driven practices require a hierarchical value model so that teams can focus on faster moving lower-level metrics that have a positive impact on the higher level business KPIs. Also, A/B testing (Liu et al., 2021; Fabijan et al., 2018a,b), and techniques such as multi-armed bandits (Issa Mattos et al., 2023) allow for experimental approaches that allow us to manage risk. Finally, we have conducted work on data infrastructure (Olsson and Bosch, 2024a) that allows companies to effectively manage the often costly and complex infrastructure surrounding the collection, storage, processing and use of data. In addition, we have worked on data pipelines (Munappy et al., 2022) as these are a key technology for data infrastructures.

6.2.4. I: Integrated cross-functional alignment

Definition: Agile never addressed the organizational issues we experienced between ‘the business,’ including product management, and R&D. Experience shows that to achieve business agility, we need to focus on cross-functional teams that incorporate all the necessary functions to respond and execute quickly. One of the main reasons is that intra-team coordination is considered an order, if not several orders, of magnitude more effective and efficient than inter-team or inter-organizational coordination. The main reason for this is that the coordination costs within a team that is often co-located, with the ability to continuously connect and generally positively inclined towards each other, is much lower than inter-team or inter-organization coordination.

Focus areas: In most companies, few topics receive as much debate as organization. To maintain conciseness and focus, we encourage companies to limit themselves to three main aspects: the nature of cross-functional teams (e.g., what competences to include in teams, how to coordinate between teams and what the scope and responsibility is for teams), steering cross-functional teams and the link between architecture and teams.

Tools: As cross-functional teams are a significant deviation from traditional hierarchical organizations, we need to use the principles of empowered organizations where individuals and teams operate without hierarchical leaders (Olsson Holmström and Bosch, 2016). This does require the hierarchical value model as it otherwise easily ends up with chaos (Olsson and Bosch, 2018b).

6.2.5. C: Continuous

Definition: Many of the case study companies still organize R&D in projects. This means that each release of a system version is often treated as a project and there is no notion of continuity around a specific product. In our experience, companies evolve from being project-centric to being product-centric and then platform-centric. This is even more important in a situation where products are expected to be continuously updated (DevOps) based on data from the field (DataOps), including the improvement of machine learning models (AI/MLOps). If the evolution of the product is supposed to be continuous, development needs to be as well.

Focus areas: In our work with the software-intensive embedded systems companies, everyone agrees in principle, many raise concerns about their specific situation, claiming that they are different and that these principles do not apply to them. There are three main arguments for why companies do not apply continuous practices. First, one frequent argument is that the customer does not want continuous deployment. Second, that the business model discourages continuous practices. Third, discuss systems engineering in the context of continuous ways of working as in many contexts, the argument is used that we cannot work continuously because of the limitations of mechanics and electronics.

Tools: One of the key challenges in the adoption of continuous practices are all the dependencies between different parts in the organization (Lwakatare et al., 2016; Bosch, 2014; Stahl et al., 2017; Mårtensson et al., 2019). Here we developed a four dimensional framework addressing business model, product upgrades, data practices and AI in a holistic model (Bosch and Olsson, 2021) that supports organizations in the transformation.

6.2.6. A: Aligned on business goals

Definition: One of the realizations that never ceases to amaze is that so few organizations are aligned on their business goals. Often, there is some high-level veneer of alignment, but when we start to drill down, it rapidly becomes clear that different departments, functions, teams and sometimes even individuals optimize for different business goals and KPIs. These can even conflict with each other, either because of local optimization or due to malicious behaviors that, in some way, are accepted within the corporate culture.

Focus areas: In order to overcome these challenges, companies need to focus on three main areas concerning business goals. First, companies need to discuss and agree on what they optimize for. Many have fallen into the worthwhile-many-versus-vital-few trap, meaning that so many aspects of the offering are considered important that it becomes meaningless to talk about focus. Instead, companies need to focus on a small “vital few” set of KPIs that everyone in the company should strive for. Second, companies need to develop mechanisms to connect teams to quantitatively defined business goals. This allows organizations to determine the success, or lack thereof, of teams. Finally, companies need to manage the evolution of business goals as companies, in different stages of maturity and in response to changes in their business ecosystem, need to evolve their strategy.

Tools: The main tool for aligning the organization as well as individual teams on common goals and desired outcomes remains value modeling. In our previous research, we have developed an approach for how to identify, agree upon and model key value factors that help teams align on the outcome they seek to achieve. By using this approach, teams and organizations can create a hierarchical model with value factors on the team level, on the product level and on the business level. In our experience, this is a rewarding way to continuously ensure that development efforts align with, and contribute to, the overall business goals. The case companies have used value modeling for years and although the creation of a hierarchical model remains a challenge due to the complexities of the systems they develop, the exercise is found rewarding as it forces an agreement on what value factors to prioritize and how to align development efforts to achieve the overall business goal (Olsson and Bosch, 2018a, 2020a; Holmström Olsson and Bosch, 2024).

6.2.7. L: Leadership

Definition: The word “Leadership” is incredibly overloaded and filled with hyperbole. And yet, companies need leadership to align teams, groups as well as the entire organization around a single goal and vision. It is incredibly difficult to navigate the organizational landscape and exercise leadership effectively as a formal leader. As many organizations are increasingly relying on informal leaders through architects, community leaders and champions, the challenge is even harder for those roles.

Focus areas: Companies need to exercise leadership through formal and informal leaders. This is concerned with three main topics. The first is organizational culture. Leaders are responsible for creating the right culture and then ensuring it is real and not a paper tiger that people pay lip service to. The second topic companies need to clarify is the notion of purpose. Especially the younger generations expect and even demand that companies have a purpose beyond earning money. It is the role of leaders to define and clarify the purpose of the company and then ensure that everyone acts in line with it. Finally, leadership often needs to deal with conflict. For all kinds of reasons, conflicts will surface in companies and failing to deal with them will often create confusion, complexities and passive-aggressive behavior that, over time, become increasingly large backpacks that the organization is dragging around. It is up to leaders to resolve conflicts and get the dissenters to the ‘disagree and commit’ point.

Tools: Leadership is a topic generally considered outside the scope of Software Engineering and we refer to the business literature. However, in our research we have studied the specific difficulties in the software-intensive industry in multiple studies including Figalist et al. (2022) and Olsson and Bosch (2020b).

6.3. Making RADICAL real

As RADICAL is a conceptual framework and our strawman proposal for what a new paradigm to product development could potentially look like, there are no companies that are currently employing the framework in its entirety. However, some of the companies involved in our research have adopted practices and tools that reflect the principles and practices of RADICAL. Below, we discuss three of these as they reflect how elements of RADICAL have started to emerge in practice. It should be noted that these are illustrative examples and not validation of the framework. What we intend is to provide an understanding for how companies explore and approach a new paradigm to address the shortcomings of their current agile ways-of-working.

During 2024, company A actively pursued an innovation strategy to exploit data together with third parties. Part of the rationale for this was to support the business strategy of the company to significantly increase its revenue from services and to complement traditional product sales. A second reason is that the company is looking at its ecosystem for novel innovations that would allow it to generate revenue from the very costly data infrastructure in the product, on premise and in the cloud, that the company has created over the last years. The approach the company took was highly aligned with RADICAL, specifically in that it is experimental, data-driven and continuous. Also, it is being driven by a cross-functional team that focused squarely on the business goals of the company.

Company G, working in telecommunications, is less constrained by certification in response to standards and has for many years adopted continuous integration and test practices. In recent years, the company has adopted DevOps meaning that any customer who wants it can get new software in its equipment every two to four weeks. The company is advanced in applying data-driven methods and carefully tracks the KPIs of its equipment to ensure that it is delivering value to customers as part of the frequent releases of new software. It operates in a continuous fashion and has found ways to use DevOps as a mechanism to reduce cost and generate additional revenue. The company was early in its industry compared to its peers in adopting this approach thanks to strong leadership by a number of key individuals.

Company B has very advanced production equipment that requires leading edge mechatronics in its systems installed at customers. As continued production is extremely important for its customers, both for predictability purposes and for reducing the risk of having to discard partially produced goods due to system failures, the company has built advanced AI driven solutions for predictive maintenance in order to make sure that the likelihood of unscheduled downtime is as small as possible. This is entirely in line with its business strategy and follows the RADICAL principles as the company, being highly responsive to its customers, is also data driven and continuous in the way it operates. Finally, interpreting and acting on the data-driven insights requires close collaboration between different disciplines.

As a common theme in these examples, the RADICAL approach allows teams and organizations to more proactively respond to change (instead of reacting as in traditional paradigms), to more effectively employ data for decision-making (compared to traditional user feedback mechanisms) and to increase alignment between engineering efforts and business (compared to engineering traditionally being all about delivering functional systems). To us, these examples represent the empirical basis for RADICAL and the ways in which product development is evolving towards what we describe as “infinite games”.

7. Discussion

Below, we discuss our results in relation to previous work. We structure our discussion around the three contributions of this paper, i.e., the challenges we identified, the principles we derived and the framework we developed.

In relation to *challenges*, prior research identifies several recurring challenges. One recurring challenge concerns the mismatch between the fast, iterative cadence of agile software teams and the slower, more sequential processes that continue to govern development once multiple teams and systems must be integrated (Eklund et al., 2014). A related challenge concerns the friction between the cultural assumptions of agile and the governance structures of large organizations. (Dikert et al., 2016) These challenges are perhaps best understood not as independent obstacles but as symptoms of a deeper incompatibility between the autonomy-oriented origin of agile practices and the hierarchical decision-making structures typical of large enterprises. This is reinforced in Paasivaara et al. (2018) in which the authors show that even a well-resourced, deliberately designed transformation initiative still required substantial coordination effort once agile introduction and organizational scaling were pursued simultaneously. In our view, this indicates that the abstract challenge categories identified in the literature manifest as concrete, sustained coordination burdens in practice. While confirming many of the earlier identified challenges, our study highlights the fact that scaling agile is not merely a matter of adding more teams, but of reconciling fundamentally incompatible cadences between fast, iterative software work and slower, more sequential systems-engineering activities. Our findings extend beyond software R&D into business models, systems engineering and organizational governance. In our view, current practices are insufficient and hence, we propose an alternative approach for future product development. As an additional reflection, there is the assumption, implicit in much of the transformation literature, that large-scale agile adoption has a stable end state. Our research as presented in e.g., Olsson and Bosch (2014) and also in this paper, complicate this assumption by showing that organizations that have successfully adopted agile at the team level still face substantial challenges when advancing toward more mature practices. This re-frames large-scale agile development as an ongoing, non-terminal process rather than a one-time transformation, and suggests that many of the challenges identified in previous literature may resurface, in modified form, at each subsequent stage of maturity rather than being resolved once and for all.

In relation to *principles*, there is limited empirical research on what a future product development paradigm could potentially look like and what requirements it must satisfy. However, approaches such as e.g., the ‘Objectives and Key Results’ (OKR) framework (Butler et al., 2024) and research on business agility e.g., Olsson and Bosch (2024b), Mathiassen and Sandberg (2021) support a similar intent, and share some of the same principles as what we present in this paper. OKR is a goal-setting and strategic alignment framework intended to drive change and focus toward ambitious, time-bound improvement in which objectives are paired with measurable key results. Similarly, much of the business agility literature emphasizes the coordination of goals, processes, and information systems across organizational levels as a precondition for effective sensing and responding to the surrounding context. Although similar in purpose, the principles we identify define the specific requirements for what a future framework must satisfy. Moreover, they are associated with tools and mechanisms for how to get there. As a common characteristic, they put an explicit emphasis on continuous, and quantitative, value modeling and development effectiveness rather than efficiency.

Finally, in relation to product development *frameworks*, a prominent body of work on the maturation of software development practice has argued that organizations do not, over time, settle into a single methodological paradigm but instead accumulate complementary practices from successive paradigms. For instance, Baskerville et al. (2011) propose that organizations moving beyond initial agile adoption tend to reintroduce classical project management structures alongside agile team-level practice. In Mathiassen and Pries-Heje (2006), the authors reach a structurally similar conclusion from a longitudinal retrospective at Ericsson, arguing that the imprints of an earlier discipline-oriented

paradigm (CMM-RUP) and a later flexibility-oriented paradigm (Agile-Lean) eventually complemented rather than displaced one another. Taken together, these accounts suggest a general pattern: methodological evolution in software organizations is additive and integrative, with earlier and later paradigms coexisting in a complementary configuration rather than one being superseded by the other. The strawman proposal that we present in this paper departs from this view. Rather than treating agile practices and RADICAL as complementary layers to be intertwined, we argue that the challenges identified in Section 4 and the principles derived in Section 5 point toward a genuine succession: RADICAL is proposed as a replacement for, rather than a supplement to, current agile practices as we know them.

To conclude, our findings validate prior work reporting on challenges in large-scale agile adoption and transformation, they extend that work by showing that in the software-intensive embedded systems domain the challenges are not confined to software R&D but extend into business models, systems engineering and organizational governance; and they challenge the prevailing assumption that these difficulties can be resolved by scaling frameworks such as SAFe or Scrum-of-Scrums, since our data indicate that the limitation originates in the assumptions of the agile paradigm itself rather than in its incomplete implementation.

8. Threats to validity

This paper rests on two empirical components with distinct validity profiles: (1) the reflective synthesis of our previously published work, and (2) the 2024 workshop series. In the section below we discuss construct, internal, external validity and reliability (Runeson and Höst, 2009) separately for each empirical component as used in this paper.

8.1. Component 1: Reflective synthesis of our previously published research

Construct validity: In this study, we present a reflective synthesis that integrates results from our prior research. All included case studies were conducted and reported following established guidelines for software engineering case study research, including explicit descriptions of context, data collection procedures, and analysis methods. The constructs introduced in those studies were therefore already empirically grounded in interviews, observations, and project artifacts as reported on in each paper.

For the *reflective synthesis* presented in this paper, the main threats are the selection of source papers, the risk of researcher bias in re-interpreting prior findings into higher-order themes, the inheritance of the primary studies' own limitations and weak traceability from synthesized constructs to the original case study evidence. To mitigate these, we worked to preserve a tight link between the new, higher-order constructs and the original case study findings. We used thematic analysis (Clarke and Braun, 2017; Cruzes et al., 2015) on the published results and maintained traceability from each synthesized theme back to the source publications and their reported findings. We documented which papers that contributed to a specific challenge and we explain how the challenges are translated into principles that are addressed in a new paradigm. This traceability is reflected in our tables and figures with the intention to enable readers to assess whether our conceptual claims are supported by the underlying empirical material.

Internal validity: To address internal validity, both researchers conducted their independent extraction and coding of findings as presented in prior publications. After this, we reconciled on potential disagreements and merged our separate interpretations into a common theme structure that lays the basis for the key challenges we present. As outlined in Section 3.4, we went from extracting findings, to identifying overarching themes, to specify and sort specific challenges in a step-by-step process and with iterations involving both researchers.

External validity: Our aim is analytical generalization in which the retrospective synthesis of multiple case studies provides a basis for

understanding patterns that recur across different organizational settings, project types, and time periods. Therefore, the challenges and the framework we present are viewed as primarily applicable in similar software engineering contexts as the ones we studied. While we believe that our findings could provide value also outside the software-intensive systems domain, we wish to emphasize that many of the challenges occur due to the nature of an embedded system. In systems without the limitations imposed by mechanics and electronics components, and where the complexity of integrating these components with software does not exist, the findings we present might be less relevant and hence, less applicable.

Reliability: Because this study is a retrospective reflective synthesis of qualitative case study research, reliability is understood as the consistency and transparency of the analytic process. We took measures to strengthen dependability by applying the same thematic analysis procedure, i.e., the six-step thematic analysis process described in Section 3.4, across the empirical material, documenting the analysis process and maintaining traceability from each synthesized theme to the original case study findings.

8.2. Component 2: On-site and online workshop series

Construct validity: We opened each workshop with a presentation in which we shared our interpretation of key concepts and terminology. In this way, we invited for a discussion among the participants, as well as between us and the participants, to ensure that we had a common understanding for the underlying concepts and terminology used in a particular workshop discussion. Our intention was to avoid misunderstandings and misinterpretations, and during our research we also shared our preliminary results on a continuous basis to allow for feedback on our interpretations.

Internal validity: We applied member checking through continuous validation with company representatives during the workshop series, we used groupthink mitigation techniques (as described in Section 3.4.2) and we shared the facilitation of the workshops between the two researchers.

External validity: Similarly to the reflective synthesis of prior publications, we acknowledge that the challenges and the framework we present are viewed as primarily applicable in similar software engineering contexts as the ones we studied. While we believe that our findings could provide value also outside the software-intensive systems domain, we wish to emphasize that many of the challenges occur due to the nature of an embedded system.

Reliability: The workshop series was conducted using a consistent procedure for recording, transcription and analysis. All group discussions were audio-recorded, transcribed using the same tool and reviewed against the recordings to ensure accuracy. For the online workshops, we used recordings and transcripts via Microsoft Teams. For the on-site workshops, we used notes and whiteboard photographs that were included in the analysis process.

9. Conclusions and future research

Agile development methods emerged as a response to the many shortcomings and limitations of previous plan-driven approaches with the intention to provide organizations with support for managing rapidly changing environments (Highsmith, 2002). Since its inception in the late 1990s, few concepts have received as much interest and agile software development has come to dominate the latter half of the past 50 years of software engineering (Hoda et al., 2018).

However, despite decades of efforts going into adopting and scaling agile practices, companies still do not experience the benefits they had expected.

In this paper, we present a reflective synthesis across a decade of collaborative research within a large research consortium. We do so by synthesizing findings across a selected set of our own prior studies into

higher-order interpretations of the challenges that software-intensive systems companies face when implementing, adopting and scaling agile practices. In addition to reflecting on our previous research, we report on a workshop series conducted in 2024 in which we refined and extended the challenges we identified and where we explored what a new paradigm could potentially look like. Despite years of efforts and huge investments being spent on adopting agile practices, our research shows that for these companies, there is a mismatch between the original agile principles and the reality of product development they face. Based on our experiences, the root of the challenges we identify is the lack of scalability. While product management, systems engineering and business models are the typical mechanisms that companies employ to scale their business, these are missing in agile. Hence, agile principles remain within the software teams but with limited impact beyond these (Olsson and Bosch, 2024b).

To address this, we present RADICAL; a conceptual framework and our strawman proposal for what a new paradigm could potentially look like. RADICAL supports not only software development, but also scaling and monetization of software-intensive systems. In our view, RADICAL is not an extension of agile, but rather a new paradigm and a different way of approaching product development and evolution. It uses several of the agile principles, but it is different in three ways; (1) *company versus software R&D*, (2) *systems versus software* and (3) *infinite vs finite games*.

The contribution of this paper is threefold. First, we identify the key challenges that companies in the software-intensive embedded systems domain experience in relation to agile practices. Second, we identify the key principles that a novel, next generation product development paradigm needs to satisfy. Third, we present RADICAL, a strawman proposal for what a new paradigm could potentially look like. Our proposal represents what we believe could succeed today's agile practices to better support all aspects of modern product development. Also, we provide three examples of how elements of RADICAL have started to emerge in practice. Our findings *validate* prior work reporting on challenges in large-scale agile adoption and transformation, they *extend* existing work by showing that in the software-intensive embedded systems domain the challenges are not confined to software R&D but extend into business models, systems engineering and organizational governance, and they *challenge* the prevailing assumption that these difficulties can be resolved by scaling frameworks such as SAFe or Scrum-of-Scrums, since our data indicate that the limitation originates in the assumptions of the agile paradigm itself rather than in its incomplete implementation.

In our future research, we aim to further explore the different aspects of product development and how the framework we present can support these. As part of this, we intend to continue our case study collaborations, expand these to involve also additional companies and

to validate the proposed framework with practitioners to get input on how to optimally support the multifaceted, and rapidly changing, product development environment they face.

CRediT authorship contribution statement

Jan Bosch: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Helena Holmström Olsson:** Writing – review & editing, Writing – original draft, Visualization, Validation, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank the workshop participants and the Software Center member companies for their valuable contributions to the research and for their long-term support and engagements in the research collaboration. Also, we thank Software Center³ for supporting this research.

The authors used Claude Opus 4.7 to improvement of the readability of the text. AI tools were not used to generate scientific content, claims, or interpretations; all conceptual contributions and the analytical findings reported in this paper are the work of the authors. The authors take full responsibility for the content of the publication.

Appendix. Challenge mapping - clusters, sub-categories, key challenges & papers/workshops

Papers may appear in more than one sub-category. Key challenges listed per sub-category are distilled from the collective contribution of the mapped papers. For each challenge category, we list the workshops where those challenges were refined and/or extended. We also list the industry domains in which a certain group of challenges are experienced (see Table A.1).

³ www.software-center.se

Table A.1
Challenge mapping-clusters, sub-categories, key challenges & papers/workshops.

Theme	Sub-category	Key Challenges	Papers/Workshops	Industry Domain(s)
Definitions and interpretations of agile	Unclear definition	- No shared, precise definition of agile, CI, CD, or DevOps - Inconsistent interpretation of what 'continuous' means in practice - Difficulty measuring adoption maturity when terminology is ambiguous - Agile defined by rituals (standups, sprints) rather than by outcomes - Business agility conflated with software delivery speed	Stahl et al. (2017), Bosch (2014), Olsson and Bosch (2014), Olsson et al. (2012), Olsson and Bosch (2024b); OS-WS1, OS-WS2	Telecommunications, Automotive, Manufacturing, Defense, Security & Surveillance, Logistics
Definitions and interpretations of agile	Regulatory constraints	- Safety and certification requirements impose phase-gate validation incompatible with iterative delivery - Regulatory approval cycles far exceed sprint cadences - Contractual obligations in B2B settings restrict the pace and scope of software changes - Traceability and auditability demands conflict with rapid, automated pipelines	Eklund et al. (2014), Lwakatare et al. (2016), Issa Mattos et al. (2023), Liu et al. (2021); OS-WS1, OS-WS3, OS-WS4	Automotive, Telecommunications, Manufacturing

(continued on next page)

Table A.1 (continued).

Theme	Sub-category	Key Challenges	Papers/Workshops	Industry Domain(s)
Definitions and interpretations of agile	Lack of adaptation to context	- Standard agile frameworks designed for pure software do not transfer to hardware-dependent or embedded contexts - Continuous deployment models assume cloud-like update mechanisms absent in embedded products - Generic agile scaling frameworks (SAFe, LeSS) fail to address domain-specific constraints - ML/AI model lifecycle does not map cleanly onto sprint-based or continuous delivery patterns - Experiment design principles from web companies cannot be directly applied to B2B or safety-critical systems	Eklund et al. (2014), Lwakatare et al. (2016), Munappy et al. (2022), Mårtensson et al. (2019), Liu et al. (2021), Issa Mattos et al. (2023); OS-WS1, OS-WS2, OS-WS3	Automotive, Telecommunication, Industrial IoT, Manufacturing, AI/ML/DL platform solutions
Definitions and interpretations of agile	Mixing goals and means	- Treating specific tools or practices as the goal rather than faster value delivery - Pursuing continuous deployment as an end in itself without linking it to business outcomes - Limited insight in how to measure quality in data-driven development - Deploying frequently without a model of what customer or business value is being created	Stahl et al. (2017), Olsson and Bosch (2020a, 2018a), Fabijan et al. (2018b), Olsson et al. (2012); OS-WS4	Automotive, Telecommunications, Manufacturing, Connectivity & Mobility
Business and monetization	Lose track of 'why' and 'what'	- Teams optimize delivery speed without a shared model of what constitutes customer or business value - Product roadmaps driven by technology capability rather than customer outcomes - Lack of value modeling causes misalignment between continuous delivery efforts and strategic business goals - Experimentation programs scale in size but not in relevance of insights - Digital product strategy decoupled from day-to-day agile execution	Olsson and Bosch (2018a, 2020a, 2024b, 2025), Fabijan et al. (2018a); OL-WS4	Automotive, Telecommunications, Manufacturing, Security & Surveillance, Defense, Connectivity & Mobility, SaaS
Business and monetization	Transactional business models	- Per-project or per-contract pricing models misalign vendor and customer incentives for continuous improvement - B2B customers expect stable, versioned releases rather than continuous updates - Revenue recognition and procurement rules designed for discrete deliverables impede subscription or continuous-update models - Digital transformation requires new monetization logic that organizations are unprepared to define or operate	Issa Mattos et al. (2023), Dakkak et al. (2022), Bosch and Olsson (2021), Olsson and Bosch (2020b, 2025); OS-WS4, OL-WS4	Telecommunications, Automotive, Defense, Healthcare, Logistics, Cybersecurity
Business and monetization	MVPs and short iterations do not work	- Field-deployed embedded products cannot be updated rapidly, making iterative hypothesis validation with real users impractical - Safety certification requirements mean even a 'minimum' viable product must pass the full validation suite - B2B customers are unwilling or unable to participate in frequent feedback loops and beta testing - Long hardware development and toolchain setup times make iteration cycles inherently long - User behavior data from embedded or industrial products is sparse, delayed, or unrepresentative	Eklund et al. (2014), Lwakatare et al. (2016), Issa Mattos et al. (2023), Dakkak et al. (2022), Backlund et al. (2014), Liu et al. (2021); OS-WS1, OS-WS2, OS-WS3, OL-WS1	Telecommunications, Automotive, Manufacturing, Web portals
Systems engineering challenges	Waterfall systems engineering mind-set	- Requirements frozen at project outset leads to changes being treated as scope creep rather than learning - System integration and testing treated as a late-stage gate rather than a continuous activity - V-model thinking separates development and verification phases in ways incompatible with CI/CD - Hardware and mechanical lead times enforce a sequential rhythm that anchors software to the same pace - Documentation-first culture prioritizes artifacts over working increments	Eklund et al. (2014), Lwakatare et al. (2016), Bosch and Olsson (2021), Olsson and Bosch (2020b), Olsson et al. (2012), Bosch (2014); OS-WS1, OS-WS2, OS-WS3, OL-WS5	Telecommunications, Automotive, Defense, Healthcare, Logistics, Cybersecurity, Power control solutions
Systems engineering challenges	Perfectionism in engineering culture	- Zero-defect norms lead to over-testing and over-specification, preventing 'good-enough' incremental releases - Reluctance to release software with known, managed technical debt - Architecture decisions driven by future-proofing rather than current need, delaying delivery - High quality bar for training data and model performance blocks iterative ML deployment - Unwillingness to accept statistical uncertainty in experiment results delays decision-making	Martini and Bosch (2019), Mårtensson et al. (2019), Eklund et al. (2014), Liu et al. (2021), Munappy et al. (2022); OS-WS2, OS-WS3	Telecommunications, Automotive, Defense, Healthcare, Logistics, Manufacturing, AI/ML/DL platform solutions

(continued on next page)

Table A.1 (continued).

Theme	Sub-category	Key Challenges	Papers/Workshops	Industry Domain(s)
Systems engineering challenges	End-to-end delivery	- Absence of a fully automated pipeline from code commit to production deployment - Manual handoffs between development, integration, test, and release teams break the continuous flow - Inadequate test automation coverage forces manual regression testing as a bottleneck - Operational tooling (monitoring, rollback, feature flags) missing or immature in embedded/industrial contexts - Data pipelines not treated as first-class delivery artifacts, creating invisible bottlenecks in ML/AI workflows	Bosch (2014), Olsson et al. (2012), Lwakatare et al. (2016), Dakkak et al. (2022), Backlund et al. (2014), Olsson and Bosch (2024a), Munappy et al. (2022); OS-WS1, OS-WS3, OS-WS4, OL-WS5	Manufacturing, Telecommunications, Automotive, Healthcare, Logistics, Defense, Connectivity & Mobility, Security & Surveillance, AI/ML/DL platform solutions
Systems engineering challenges	Difficulties in decoupling	- Hardware and software release cycles tightly coupled, preventing independent software updates - Monolithic architectures make it impossible to deploy individual components independently - Shared platform dependencies between products or business units create mutual blocking - Technical debt and architectural erosion increase coupling over time, progressively degrading deployment frequency - Legacy integration contracts between systems prevent modular evolution	Eklund et al. (2014), Lwakatare et al. (2016), Mårtensson et al. (2019), Martini and Bosch (2019), Bosch and Olsson (2021), Olsson and Bosch (2020b); OS-WS1, OS-WS2, OL-WS5	Automotive, Telecommunications, Manufacturing, Defense, Healthcare, Power control solutions, Logistics
Systems engineering challenges	Lack of trust	- Engineers distrust automated test suites to catch regressions, defaulting to manual verification - Managers and stakeholders distrust AI-generated analytics insights, preferring manual analysis - Statistical validity of A/B test results questioned, leading to decision paralysis or overrides - Data quality concerns prevent teams from acting on continuously collected telemetry - Lack of transparency in ML model behavior undermines confidence in AI-assisted tooling	Figalist et al. (2022), Munappy et al. (2022), Olsson and Bosch (2024a), Fabijan et al. (2018a,b), Issa Mattos et al. (2023); OS-WS3, OS-WS4	Telecommunications, Manufacturing, Healthcare, AI/ML/DL platform solutions, SaaS
Organization, culture and ways-of-working	Mismatch with rest of the company	- Agile software teams operate at a different cadence than sales, procurement, legal, and hardware departments - Customer-facing teams (support, account management) are not equipped to handle continuously changing products - Marketing and roadmap communication based on discrete release milestones incompatible with continuous delivery - Organizational structures reward functional silos rather than end-to-end product flow	Olsson and Bosch (2014), Olsson et al. (2013), Olsson Holmström and Bosch (2016), Bosch and Olsson (2021), Olsson and Bosch (2020b, 2025), Dakkak et al. (2022); OS-WS1, OS-WS3, OS-WS4, OL-WS2	Telecommunications, Manufacturing, Automotive, Healthcare, Defense, Logistics, Power control solutions, SaaS
Organization, culture and ways-of-working	Mismatch in governance	- Annual budgeting cycles incompatible with iterative, value-based investment decisions - Portfolio-level governance designed for project milestones cannot accommodate rolling product backlogs - Risk and compliance reporting frameworks require predictability that agile delivery intentionally avoids - Empowered autonomous teams create accountability gaps in traditional governance structures - Experiment-driven decision-making challenges authority of senior stakeholders	Olsson and Bosch (2024b), Olsson et al. (2013), Olsson and Bosch (2018b), Olsson Holmström and Bosch (2016), Olsson and Bosch (2025), Fabijan et al. (2018a); OL-WS3	Automotive, Telecommunications
Organization, culture and ways-of-working	Interdependencies	- Cross-team dependencies on shared components, infrastructure, or data sources create coordination bottlenecks - Hardware–software co-development requires synchronization that conflicts with independent team autonomy - Shared platform teams become bottlenecks when multiple product teams request changes simultaneously - Supplier and partner dependencies introduce external constraints on release schedules - Architectural interdependencies force implicit coordination that erodes team-level agility	Eklund et al. (2014), Olsson et al. (2013), Lwakatare et al. (2016), Bosch and Olsson (2021), Olsson and Bosch (2020b), Martini and Bosch (2019); OS-WS3, OL-WS5	Telecommunications, Manufacturing, Automotive, Healthcare, Defense, Logistics, Power control solutions
Organization, culture and ways-of-working	Customization instead of configuration	- Teams build bespoke tooling and pipelines instead of adopting and configuring standard platforms - Custom data pipelines and ML infrastructure are hard to maintain and scale as product evolves - Proprietary internal tooling creates knowledge silos and complicates onboarding - Over-tailored processes make it difficult to adopt industry best practices or common patterns	Figalist et al. (2022), Olsson and Bosch (2024a), Munappy et al. (2022), Holmström Olsson and Bosch (2024)	Manufacturing, Healthcare, Defense, Power control solutions, Automotive, Security & Surveillance, AI/ML/DL platform solutions

(continued on next page)

Table A.1 (continued).

Theme	Sub-category	Key Challenges	Papers/Workshops	Industry Domain(s)
Organization, culture and ways-of-working	Lack of short feedback cycles	- Long release cycles mean customer feedback arrives too late to influence the next iteration - Infrequent deployments make it impossible to validate hypotheses at the pace required for continuous learning - Slow automated test suites delay pipeline execution and reduce commit frequency - Organizational processes (change management boards, release approvals) add latency between decision and deployment - Experiment infrastructure not mature enough to provide actionable results at the speed of development	Olsson et al. (2012), Bosch (2014), Olsson and Bosch (2014), Olsson et al. (2013), Fabijan et al. (2018a); OS-WS1, OS-WS2, OS-WS3, OS-WS4, OL-WS1	Automotive, Manufacturing, Healthcare, Defense, Telecommunications, Security & Surveillance, SaaS; Web portals
Organization, culture and ways-of-working	Small improvements – commoditization	- Relentless focus on small, safe increments prevents the step-change innovations needed for competitive differentiation - Continuous delivery culture systematically de-risks change, eroding appetite for transformational bets - Product strategy reduced to backlog grooming, losing sight of longer-term market positioning - Feature parity with competitors achieved through iteration, but distinctive value propositions are not created	Olsson and Bosch (2024b, 2025, 2020a, 2018a, 2020b); OS-WS4	Automotive, Manufacturing, Healthcare, Defense, Telecommunications, Security & Surveillance, Power control solutions
Organization, culture and ways-of-working	No scaling beyond teams	- Agile practices that work at team level break down when applied across programmes or portfolios - Non-hierarchical, self-organizing structures lack the coordination mechanisms needed at enterprise scale - Team empowerment without strategic alignment leads to local optimization and global incoherence - Data and experimentation capabilities built within teams are not systematically shared or scaled - Governance and tooling designed for small teams does not support large-scale continuous delivery	Olsson et al. (2013), Olsson Holmström and Bosch (2016), Olsson and Bosch (2018b), Eklund et al. (2014), Fabijan et al. (2018a), Holmström Olsson and Bosch (2024); OS-WS1, OS-WS2, OL-WS2	Manufacturing, Telecommunications, Automotive, Defense, Security & Surveillance

Data availability

The authors do not have permission to share data.

References

- Al-Saqqah, S., Sawalha, S., AbdelNabi, H., 2020. Agile software development: Methodologies and trends.. *Int. J. Interact. Mob. Technol.* 14 (11).
- AlMutairi, A.M., Qureshi, M.R.J., 2015. The proposal of scaling the roles in scrum of scrums for distributed large projects. *J. Inf. Technol. Comput. Sci. (IJTCS)* 7 (8), 68–74.
- Backlund, E., Bolle, M., Tichy, M., Olsson, H.H., Bosch, J., 2014. Automated user interaction analysis for workflow-based web portals. In: *Software Business. Towards Continuous Value Delivery: 5th International Conference, ICSOB 2014, Paphos, Cyprus, June 16–18, 2014. Proceedings 5*. Springer, pp. 148–162.
- Baskerville, R., Pries-Heje, J., Madsen, S., 2011. Post-agility: What follows a decade of agility? *Inf. Softw. Technol.* 53 (5), 543–555.
- Bianchi, M., Marzi, G., Zollo, L., et al., 2018. Beyond customer needs software development: an empirical investigation of its forms and individual-level causes. In: *International Product Development Management Conference 2018. IPDMA*.
- Birkinshaw, J., 2019. What to expect from agile. *MIT Sloan Manag. Rev. Spec. Collect. Stay. Agil.* 1, 8–11.
- Bosch, J., 2014. Continuous software engineering: An introduction. In: *Continuous Software Engineering*. Springer, pp. 3–13.
- Bosch, J., Olsson, H.H., 2021. Digital for real: A multicase study on the digital transformation of companies in the embedded systems domain. *J. Softw.: Evol. Process.* 33 (5), e2333.
- Butler, J.L., Zimmermann, T., Bird, C., 2024. Objectives and key results in software teams: Challenges, opportunities and impact on development. In: *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*. pp. 358–368.
- Campbell, S., Greenwood, M., Prior, S., Shearer, T., Walkem, K., Young, S., Bywaters, D., Walker, K., 2020. Purposive sampling: Complex or simple? Research case examples. *J. Res. Nurs.* 25 (8), 652–661.
- Carse, J., 2011. Finite and Infinite Games. Simon and Schuster.
- Chandra Misra, S., Kumar, V., Kumar, U., 2010. Identifying some critical changes required in adopting agile practices in traditional software development projects. *Int. J. Qual. Reliab. Manag.* 27 (4), 451–474.
- Clarke, V., Braun, V., 2017. Thematic analysis. *J. Posit. Psychol.* 12 (3), 297–298.
- Conboy, K., 2009. Agility from first principles: Reconstructing the concept of agility in information systems development. *Inf. Syst. Res.* 20 (3), 329–354.
- Cruzes, D.S., Dybå, T., Runeson, P., Höst, M., 2015. Case studies synthesis: a thematic, cross-case, and narrative synthesis worked example. *Empir. Softw. Eng.* 20 (6), 1634–1665.
- Dakkak, A., Munappy, A.R., Bosch, J., Olsson, H.H., 2022. Customer support in the era of continuous deployment: A software-intensive embedded systems case study. In: *2022 IEEE 46th Annual Computers, Software, and Applications Conference. COMPSAC, IEEE*, pp. 914–923.
- Dikert, K., Paasivaara, M., Lassenius, C., 2016. Challenges and success factors for large-scale agile transformations: A systematic literature review. *J. Syst. Softw.* 119, 87–108.
- Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B., 2012. A decade of agile methodologies: Towards explaining agile software development.
- Dobson, P.J., 2001. Longitudinal case research: A critical realist perspective. *Syst. Pr. Action Res.* 14, 283–296.
- Dougllass, B.P., 2015. *Agile Systems Engineering*. Morgan Kaufmann.
- Dove, R., 2005. Fundamental principles for agile systems engineering. In: *Conference on Systems Engineering Research (CSER)*. Stevens Institute of Technology, Hoboken, NJ.
- Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.* 50 (9–10), 833–859.
- Edison, H., Wang, X., Conboy, K., 2021. Comparing methods for large-scale agile software development: A systematic literature review. *IEEE Trans. Softw. Eng.* 48 (8), 2709–2731.
- Eklund, U., Berger, C., 2017. Scaling agile development in mechatronic organizations—a comparative case study. In: *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE, pp. 173–182.
- Eklund, U., Holmström Olsson, H., Strøm, N.J., 2014. Industrial challenges of scaling agile in mass-produced embedded systems. In: *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation: XP 2014 International Workshops, Rome, Italy, May 26–30, 2014, Revised Selected Papers 15*. Springer, pp. 30–42.
- Etikan, I., Musa, S.A., Alkassim, R.S., et al., 2016. Comparison of convenience sampling and purposive sampling. *Am. J. Theor. Appl. Stat.* 5 (1), 1–4.
- F. Tripp, J., Armstrong, D.J., 2018. Agile methodologies: organizational adoption motives, tailoring, and performance. *J. Comput. Inf. Syst.* 58 (2), 170–179.
- Fabijan, A., Dmitriev, P., McFarland, C., Vermeer, L., Holmström Olsson, H., Bosch, J., 2018a. Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies. *J. Softw.: Evol. Process.* 30 (12), e2113.
- Fabijan, A., Dmitriev, P., Olsson, H.H., Bosch, J., 2018b. The online controlled experiment lifecycle. *IEEE Softw.* 37 (2), 60–67.

- Figalist, I., Elsner, C., Bosch, J., Olsson, H.H., 2019. Scaling agile beyond organizational boundaries: Coordination challenges in software ecosystems. In: *Agile Processes in Software Engineering and Extreme Programming: 20th International Conference, XP 2019, Montréal, QC, Canada, May 21–25, 2019, Proceedings* 20. Springer International Publishing, pp. 189–206.
- Figalist, I., Elsner, C., Bosch, J., Olsson, H.H., 2022. Breaking the vicious circle: A case study on why AI for software analytics and business intelligence does not take off in practice. *J. Syst. Softw.* 184, 111135.
- Fowler, M., Highsmith, J., et al., 2001. The agile manifesto. *Softw. Dev.* 9 (8), 28–35.
- Gren, L., Torkar, R., Feldt, R., 2015. The prospects of a quantitative measurement of agility: A validation study on an agile maturity model. *J. Syst. Softw.* 107, 38–49.
- Habibullah, K.M., Gay, G., Horkoff, J., 2024a. A framework for managing quality requirements for machine learning-based software systems. In: *International Conference on the Quality of Information and Communications Technology*. Springer, pp. 3–20.
- Habibullah, K.M., Heyn, H.-M., Gay, G., Horkoff, J., Knauss, E., Borg, M., Knauss, A., Sivencrona, H., Li, P.J., 2024b. Requirements and software engineering for automotive perception systems: an interview study. *Requir. Eng.* 1–24.
- Hanssen, G.K., Stålhane, T., Myklebust, T., 2018. *SafeScrum®-Agile development of safety-critical software*. Springer.
- Highsmith, J., 2002. *Agile Software Development Ecosystems*. Addison-Wesley.
- Hoda, R., Salleh, N., Grundy, J., 2018. The rise and evolution of agile software development. *IEEE Softw.* 35 (5), 58–63.
- Holmström Olsson, H., Bosch, J., 2024. How to get good at data: 5 steps. In: *Proceedings of the 7th ACM/IEEE International Workshop on Software-Intensive Business*. pp. 32–39.
- Iivari, J., Iivari, N., 2011. The relationship between organizational culture and the deployment of agile methods. *Inf. Softw. Technol.* 53 (5), 509–520.
- Issa Mattos, D., Dakkak, A., Bosch, J., Olsson, H.H., 2023. The HURRIER process for experimentation in business-to-business mission-critical systems. *J. Softw.: Evol. Process.* 35 (5), e2390.
- Jones, L.G., Soule, A.L., 2002. *Software Process Improvement and Product Line Practice: CMMI and the Framework for Software Product Line Practice*. Carnegie Mellon University, Software Engineering Institute.
- Kasauli, R., Knauss, E., Horkoff, J., Liebel, G., de Oliveira Neto, F.G., 2021. Requirements engineering challenges and practices in large-scale agile system development. *J. Syst. Softw.* 172, 110851.
- Kasauli, R., Knauss, E., Kanagaw, B., Nilsson, A., Calikli, G., 2018. Safety-critical systems and agile development: A mapping study. In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications*. SEAA, IEEE, pp. 470–477.
- Kišš, F., Rossi, B., 2018. Agile to lean software development transformation: A systematic literature review. In: *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, pp. 969–973.
- Kotter, J., Heskett, J., 1992. Corporate culture and performance.
- Kurapati, N., Manyam, V.S.C., Petersen, K., 2012. Agile software development practice adoption survey. In: *Agile Processes in Software Engineering and Extreme Programming: 13th International Conference, XP 2012, Malmö, Sweden, May 21–25, 2012*. Proceedings 13. Springer, pp. 16–30.
- Laanti, M., Salo, O., Abrahamsson, P., 2011. Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Inf. Softw. Technol.* 53 (3), 276–290.
- Liu, Y., Mattos, D.I., Bosch, J., Olsson, H.H., Lantz, J., 2021. Bayesian propensity score matching in automotive embedded software engineering. In: *2021 28th Asia-Pacific Software Engineering Conference*. APSEC, IEEE, pp. 233–242.
- Lwakatare, L.E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H.H., Bosch, J., Oivo, M., 2016. Towards DevOps in the embedded systems domain: Why is it so hard? In: *2016 49th Hawaii International Conference on System Sciences (Hicss)*. IEEE, pp. 5437–5446.
- Mårtensson, T., Ståhl, D., Martini, A., Bosch, J., 2019. Continuous architecture: Towards the goldilocks zone and away from vicious circles. In: *2019 IEEE International Conference on Software Architecture*. ICSA, IEEE, pp. 131–140.
- Martini, A., Bosch, J., 2019. Architectural technical debt in embedded systems. *Syst. Eng. Fourth Ind. Revolut.* 77–103.
- Mathiassen, L., Pries-Heje, J., 2006. Business agility and diffusion of information technology.
- Mathiassen, L., Sandberg, A., 2013. How a professionally qualified doctoral student bridged the practice-research gap: A confessional account of collaborative practice research. *Eur. J. Inf. Syst.* 22 (4), 475–492.
- Mathiassen, L., Sandberg, A., 2021. Complementary technology traces in software practice: A retrospective reflection over sixteen years of evolution at ericsson. *Eur. J. Inf. Syst.* 30 (6), 642–658.
- Maxwell, J.A., 2012. *Qualitative Research Design: An Interactive Approach*. Sage publications.
- McMackin, J., Heffernan, M., 2021. Agile for HR: fine in practice, but will it work in theory? *Hum. Resour. Manag. Rev.* 31 (4), 100791.
- Misra, S.C., Kumar, V., Kumar, U., 2009. Identifying some important success factors in adopting agile software development practices. *J. Syst. Softw.* 82 (11), 1869–1890.
- Mohamed, B., Moselhi, O., 2019. A framework for utilization of agile management in construction management. In: *Can Soc Civ Eng Annu Conf, CSCE: CON21*. pp. 1–10.
- Munappy, A.R., Bosch, J., Olsson, H.H., Arpteg, A., Brinne, B., 2022. Data management for production quality deep learning models: Challenges and solutions. *J. Syst. Softw.* 191, 111359.
- Olsson, H.H., Alahyari, H., Bosch, J., 2012. Climbing the "stairway to heaven"—a multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, pp. 392–399.
- Olsson, H.H., Bosch, J., 2014. Towards agile and beyond: an empirical account on the challenges involved when advancing software development practices. In: *Agile Processes in Software Engineering and Extreme Programming: 15th International Conference, XP 2014, Rome, Italy, May 26–30, 2014*. Proceedings 15. Springer, pp. 327–335.
- Olsson, H.H., Bosch, J., 2018a. Make up your mind: towards a comprehensive definition of customer value in large scale software development. *Clei Electron. J.* 21 (1), 2–1.
- Olsson, H.H., Bosch, J., 2018b. Singing the praise of empowerment: or paying the cost of chaos. In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications*. SEAA, IEEE, pp. 17–21.
- Olsson, H.H., Bosch, J., 2020a. The five purposes of value modeling. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications*. SEAA, IEEE, pp. 110–119.
- Olsson, H.H., Bosch, J., 2020b. Going digital: Disruption and transformation in software-intensive embedded systems ecosystems. *J. Softw.: Evol. Process.* 32 (6), e2249.
- Olsson, H.H., Bosch, J., 2024a. Dealing with data: Bringing order to chaos. In: *Proceedings SEAA 2024*.
- Olsson, H.H., Bosch, J., 2024b. Towards business agility 2.0. In: *International Conference on Digital Product Management*. Springer, pp. 1–14.
- Olsson, H.H., Bosch, J., 2025. Strategic digital product management: Nine approaches. *Inf. Softw. Technol.* 177, 107594.
- Olsson, H., Sandberg, A., Bosch, J., Alahyari, H., 2013. Scale and responsiveness in large-scale software development. *IEEE Softw.* 31 (5), 87–93.
- Olsson Holmström, H., Bosch, J., 2016. No more bosses?: A multi-case study on the emerging use of non-hierarchical principles in large-scale software development. In: *17th International Conference on Product-Focused Software Process Improvement, Trondheim, Norway (November 22nd–24th 2016)*. Springer, pp. 86–101.
- Paasivaara, M., Behm, B., Lassenius, C., Hallikainen, M., 2018. Large-scale agile transformation at ericsson: a case study. *Empir. Softw. Eng.* 23, 2550–2596.
- Paasivaara, M., Lassenius, C., Heikkilä, V.T., 2012. Inter-team coordination in large-scale globally distributed scrum: Do scrum-of-scrums really work? In: *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. pp. 235–238.
- Patton, M.Q., 1999. Enhancing the quality and credibility of qualitative analysis. *Health Serv. Res.* 34 (5 Pt 2), 1189.
- Perkusich, M., e Silva, L.C., Costa, A., Ramos, F., Saraiva, R., Freire, A., Dilenzo, E., Dantas, E., Santos, D., Gorgônio, K., et al., 2020. Intelligent software engineering in the context of agile software development: A systematic literature review. *Inf. Softw. Technol.* 119, 106241.
- Royce, W., 2002. CMM vs. CMMI: From conventional to modern software management. *Ration. Edge* 2, 82.
- Ruk, S.A., Khan, M.F., Khan, S.G., Zia, S.M., 2019. A survey on adopting agile software development: issues & its impact on software quality. In: *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences*. ICETAS, IEEE, pp. 1–5.
- Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. *Empir. Softw. Eng.* 14, 131–164.
- Russo, D., 2021. The agile success model: a mixed-methods study of a large-scale agile transformation. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* 30 (4), 1–46.
- Schwaber, K., 1997. *Scrum development process*. In: *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings 16 October 1995, Austin, Texas*. Springer, pp. 117–134.
- Setiawati, R., Eve, J., Syavira, A., Ricardianto, P., Endri, E., et al., 2022. The role of information technology in business agility: Systematic literature review. *Calitatea* 23 (189), 144–149.
- Shmueli, O., Ronen, B., 2017. Excessive software development: Practices and penalties. *Int. J. Proj. Manage.* 35 (1), 13–27.
- Stahl, D., Martensson, T., Bosch, J., 2017. Continuous practices and devops: beyond the buzz, what does it all mean? In: *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications*. SEAA, IEEE, pp. 440–448.
- Stray, V., Gundelsby, J.H., Ulfesnes, R., Brede Moe, N., 2022. How agile teams make objectives and key results (OKRs) work. In: *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering*. pp. 104–109.

- Turetken, O., Stojanov, I., Trienekens, J.J., 2017. Assessing the adoption level of scaled agile development: a maturity model for scaled agile framework. *J. Softw.: Evol. Process.* 29 (6), e1796.
- Turner, R., 2007. Toward agile systems engineering processes. *Crosstalk. J. Déf. Softw. Eng.* 11–15.
- Walsham, G., 1995. Interpretive case studies in IS research: nature and method. *Eur. J. Inf. Syst.* 4 (2), 74–81.
- Williams, L., Cockburn, A., 2003. Agile software development: It's about feedback and change. *Computer* 36 (6), 39–43.
- Yin, R.K., 1994. Discovering the future of the case study. *Method in evaluation research. Eval. Pr.* 15 (3), 283–290.
- Yin, R.K., 2003. Designing case studies. *Qual. Res. Methods* 5 (14), 359–386.
- Yürüm, O.R., Demirörs, O., Rabhi, F., 2018. A comprehensive evaluation of agile maturity self-assessment surveys. In: *International Conference on Software Process Improvement and Capability Determination*. Springer, pp. 300–315.