Darja Šmite · Eduardo Guerra · Xiaofeng Wang · Michele Marchesi · Peggy Gregory (Eds.)

LNBIP 512

Agile Processes in Software Engineering and Extreme Programming

25th International Conference on Agile Software Development
XP 2024, Bozen-Bolzano, Italy, June 4–7, 2024
Proceedings

Springer

Lecture Notes in Business Information Processing 512

LNBIP reports state-of-the-art results in areas related to business information systems and industrial application software development – timely, at a high level, and in both printed and electronic form.

The type of material published includes

- Proceedings (published in time for the respective event)
- Postproceedings (consisting of thoroughly revised and/or extended final papers)
- Other edited monographs (such as, for example, project reports or invited volumes)
- Tutorials (coherently integrated collections of lectures given at advanced courses, seminars, schools, etc.)
- Award-winning or exceptional theses

LNBIP is abstracted/indexed in DBLP, EI and Scopus. LNBIP volumes are also submitted for the inclusion in ISI Proceedings.

Darja Šmite · Eduardo Guerra · Xiaofeng Wang ·
Michele Marchesi · Peggy Gregory
Editors

Agile Processes in Software Engineering and Extreme Programming

25th International Conference on Agile Software Development
XP 2024, Bozen-Bolzano, Italy, June 4–7, 2024
Proceedings

Springer

Editors
Darja Šmite
Blekinge Institute of Technology
Karlskrona, Sweden

Xiaofeng Wang
Free University of Bozen-Bolzano
Bozen-Bolzano, Italy

Peggy Gregory
University of Glasgow
Glasgow, UK

Eduardo Guerra
Free University of Bozen-Bolzano
Bozen-Bolzano, Italy

Michele Marchesi
Emeritus Professor in Software Engineering,
University of Cagliari
Cagliari, Italy

ISSN 1865-1348 ISSN 1865-1356 (electronic)
Lecture Notes in Business Information Processing
ISBN 978-3-031-61153-7 ISBN 978-3-031-61154-4 (eBook)
https://doi.org/10.1007/978-3-031-61154-4

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Preface

Welcome to this special anniversary volume of research proceedings of XP 2024, the 25th International Conference on Agile Software Development, organized in picturesque Bolzano, Italy, from June 4th to 7th, 2024.

For over a quarter of the century, the XP conference has served as the premier focus for researchers, practitioners, educators, and visionaries. Originating with a focus on eXtreme Programming, this conference has evolved to embrace the entire spectrum of modern Agile methodologies and the broader landscape of Agility, Agile, and Lean thinking, reaching beyond the boundaries of the software industry.

The theme of this special anniversary issue, "Reflect, Adapt, Envision," prompts the Agile community to pause, reflect on the progress made, and envision future directions for both research and practice in the field.

In this volume, we proudly present 10 full papers and 2 short papers that exemplify the ideas shaping the present state of our community. All papers underwent a thorough double-blind review process with 3 reviewers and included papers were selected from a total of 32 submissions.

The selected conference papers address a wide range of topics and tackle diverse practical problems relevant to the Agile community, such as scaling Agile practices for large development environments, facilitating Agile transformations, advancing Agile product management, and supporting education and training. Additionally, this volume features insights into the emergent topics of Artificial Intelligence (AI) and Sustainability. We trust that the latest research will inspire fruitful discussions and ignite further initiatives aimed at the adoption, refinement, and advancement of Agile software development approaches.

Our sincerest gratitude goes out to all the authors whose contributions, interest, and dedication have been integral to the success of the XP conferences. We also extend our thanks to the reviewers for their passion and invaluable feedback, which have contributed to the continuous development of the field. Special appreciation is owed to all the organizers, session chairs, and participants. Finally, we express our deepest gratitude to the XP Conference Steering Committee and the Agile Alliance for the support that helps to promote the XP conference as a leading forum for the Agile community.

June 2024

Darja Šmite
Eduardo Guerra
Peggy Gregory

Organization

Conference Co-chairs

Xiaofeng Wang	Free University of Bozen-Bolzano, Italy
Michele Marchesi	Cagliari University, Italy

Program Co-chairs

Darja Šmite	Blekinge Institute of Technology, Sweden
Eduardo Guerra	Free University of Bozen-Bolzano, Italy

Publication Chair

Peggy Gregory	University of Glasgow, UK

Program Committee

Ademar Aguiar	University of Porto, Portugal
Abdullah Aldaeej	University of Maryland, USA
Scott Ambler	SA+A, Canada
Craig Anslow	Victoria University of Wellington, New Zealand
Hubert Baumeister	Technical University of Denmark, Denmark
Jan Bosch	Chalmers University of Technology, Sweden
Frank Buschmann	Siemens AG, Germany
Filipe Correia	University of Porto, Portugal
Daniele Cruzes	Norwegian University of Science and Technology, Norway
Tiago Silva Da Silva	Universidade Federal de São Paulo, Brazil
Torgeir Dingsøyr	Norwegian University of Science and Technology, Norway
Yael Dubinsky	Kinneret Academic College, Israel
Jutta Eckstein	Independent, Germany
Henry Edison	Blekinge Institute of Technology, Sweden
Juan Garbajosa	Universidad Politécnica de Madrid, Spain
Alfredo Goldman	University of São Paulo, Brazil
Peggy Gregory	University of Glasgow, UK

Helena Holmström Olsson	University of Malmö, Sweden
Kiyoshi Honda	Osaka Institute of Technology, Japan
Eriks Klotins	Blekinge Institute of Technology, Sweden
Fabio Kon	University of São Paulo, Brazil
Marco Kuhrmann	Reutlingen University, Germany
Casper Lassenius	Aalto University, Finland
Ville Leppänen	University of Turku, Finland
Lech Madeyski	Wroclaw University of Science and Technology, Poland
Sabrina Marczak	PUCRS, Brazil
Antonio Martini	University of Oslo, Norway
Frank Maurer	University of Calgary, Canada
Jorge Melegati	Free University of Bozen-Bolzano, Italy
Tommi Mikkonen	University of Jyväskylä, Finland
Alok Mishra	Atilim University, Turkey
Nils Brede Moe	SINTEF, Norway
Jürgen Münch	Reutlingen University, Germany
Anh Nguyen Duc	University College of Southeast Norway, Norway
Maria Paasivaara	LUT University & Aalto University, Finland
Cécile Péraire	Carnegie Mellon University, USA
Rafael Prikladnicki	PUCRS, Brazil
Adam Przybylek	Gdansk University of Technology, Poland
Pilar Rodríguez	Universidad Politécnica de Madrid, Spain
Helen Sharp	Open University, UK
Christoph J. Stettina	Leiden University/Centre for Innovation, The Netherlands
Paolo Tell	IT University of Copenhagen, Denmark
Nirnaya Tripathi	University of Oulu, Finland
John F. Tripp	Clemson University, USA
Rini van Solingen	Delft University of Technology, The Netherlands
Joost Visser	Leiden University, The Netherlands
Stefan Wagner	University of Stuttgart, Germany
Hironori Washizaki	Waseda University, Japan
Agustin Yague	Universidad Politécnica de Madrid, Spain
Luciana Zaina	Universidade Federal de São Carlos, Brazil
Franz Zieris	Blekinge Institute of Technology, Sweden

Steering Committee

Hubert Baumeister	Technical University of Denmark, Denmark
François Coallier	École de Technologie Supérieure, Canada

Jutta Eckstein	Independent, Germany
Hendrik Esser	Ericsson, Germany
Teresa Foster	Agile Alliance, USA
Juan Garbajosa	Universidad Politécnica de Madrid, Spain
Peggy Gregory (Chair)	University of Glasgow, UK
Wouter Lagerweij	Lagerweij Consultancy, The Netherlands
Maria Paasivaara	LUT University & Aalto University, Finland
Viktoria Stray	University of Oslo, Norway
Xiaofeng Wang	Free University of Bozen-Bolzano, Italy

Sponsoring Organization

Teresa Foster	Agile Alliance, USA

Contents

Agile at Scale

Investigating Communities of Practice in Large-Scale Agile Software Development: An Interview Study . . . 3
Franziska Tobisch, Johannes Schmidt, and Florian Matthes

Slack Use in Large-Scale Agile Organizations: ESN Tools as Catalysts for Alignment? . . . 20
Viktoria Stray and Astri Barbala

Coordination in Agile Product Areas: A Case Study from a Large FinTech Organization . . . 36
Marthe Berntzen, Silje Alette Engdal, Maja Gellein, and Nils Brede Moe

Software Product Management in Large-Scale Agile . . . 53
Nils Brede Moe, Marthe Berntzen, Astri Barbala, and Viktoria Stray

Investigating Effort Estimation in a Large-Scale Agile ERP Transformation Program . . . 70
Franziska Tobisch, Karla Weigelt, Pascal Philipp, and Florian Matthes

Value and Quality in Agile

The Current State of Operationalizing Value by Dutch Product Owners in Agile Software Development . . . 89
Erik van Daalen and Rini van Solingen

Impact of the Kanban Maturity Model on a Team's Agile Transformation: Tripling Throughput and Elevating Quality in Three Months . . . 107
Jacek Trzesicki, Krzysztof Marek, and Adam Przybylek

LLM-Based Agents for Automating the Enhancement of User Story Quality: An Early Report . . . 117
Zheying Zhang, Maruf Rayhan, Tomas Herda, Manuel Goisauf, and Pekka Abrahamsson

People and Teams in Agile

Comparing Stability and Sustainability in Agile Systems 129
Robert Healy, Kieran Conboy, Tapajit Dey, Edwin Lewzey, and Brian Fitzgerald

Onboarding for an Agile Software Development Company 144
Tomi Enberg, Sari Alander, and Maria Paasivaara

Exploring Human-AI Collaboration in Agile: Customised LLM Meeting Assistants 163
Beatriz Cabrero-Daniel, Tomas Herda, Victoria Pichler, and Martin Eder

The Role of Team Composition in Agile Software Development Education: A Gendered Perspective 179
Gyda Elisa Sæter, Viktoria Stray, Steffen Almås, and Yngve Lindsjørn

Author Index 197

Agile at Scale

Investigating Communities of Practice in Large-Scale Agile Software Development: An Interview Study

Franziska Tobisch(✉), Johannes Schmidt, and Florian Matthes

TUM School of Computation, Information and Technology, Department of Computer Science, Technical University of Munich, Munich, Germany
{franziska.tobisch,johannes.schmidt,florian.matthes}@tum.de

Abstract. Nowadays, responsiveness is essential to be competitive, particularly in software development. Traditional methods face limitations in meeting this demand for agility, which led to the rise of agile practices. Inspired by their success in small projects, organizations have begun to use agile methods in larger contexts. However, scaling agile practices introduces complexities and requires coordinating teams, managing dependencies, and collaboration. Communities of Practices (CoPs) are argued to address these issues and support organizations in adopting agile methods at scale. Still, empirical insights into the establishment of CoPs in scaled agile settings are limited. This study fills this gap by conducting expert interviews, exploring why organizations applying agile methods at scale adopt CoPs, and examining their characteristics. Our key findings include that, next to benefit from known advantages of CoPs, like knowledge sharing, organizations establish them to foster empowerment, strengthen alignment, and drive their agile transformation. Moreover, CoPs focus not only on agile but also on classical themes such as architecture. Communities are not necessarily established bottom-up but are often initiated by management, e.g., to empower employees. In general, CoPs are accepted by management and play an essential role in decision-making.

Keywords: Agile software development · Large-scale agile · Communities of Practice

1 Introduction

Today's fast-changing business environment forces companies to react quickly to stay competitive [34]. In our digitalized world, software development is particularly impacted by this need for agility [6]. While traditional development methods increasingly reach their limits [6], agile practices and frameworks, like Scrum [26], started spreading widely due to their potential benefits regarding responsiveness and resilience to change [2]. Motivated by the success of agile

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 3–19, 2024.
https://doi.org/10.1007/978-3-031-61154-4_1

methods in small projects, organizations have begun to use them in a larger context [2,3], including the joint development of software by multiple agile teams and the adoption of agile methods and principles across the organization [4].

The complexity of scaling agile methods and resulting changes in the organizational structure, e.g., toward cross-functional teams, lead to challenges [2,3]. Besides efficiently coordinating multiple teams, organizations struggle with undiscovered dependencies, redundant work, silo thinking, lacking collaboration, and inconsistent agile practices [2,3]. Also, knowledge and experience regarding agile are often lacking [2,3]. To avoid the mentioned problems, concepts for cross-organizational communication and collaboration [2,4], continuous improvement [2], and alignment [3] are crucial. Several scaling agile frameworks recommend Communities of Practice (CoPs) [5,12,14,24], groups of people with a common interest who regularly exchange ideas, share experiences and knowledge, and learn from each other [35]. Despite extensive research in other contexts, limited empirical insights about CoPs exist in scaled agile settings. By now, researchers have mainly studied individual communities in-depth or analyzed communities in one organization, not providing a broad overview of communities established in practice. To fill this need for further research [3,20,21,33], we conducted a large expert interview study investigating why organizations applying agile methods at scale establish CoPs and what characteristics these communities have. We defined the following research questions (RQs) for our study:

RQ1: What are reasons for establishing CoPs in large-scale agile environments?
RQ2: What CoPs exist in organizations applying agile methods at scale?

2 Background and Related Work

CoPs are *"groups of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise by interacting on an ongoing basis"* [35]. Those groups can exist in various contexts (e.g., organizations, governments, and education) [35]. A CoP has three crucial characteristics [35]: (1) Domain, the shared interest of its members; (2) Community, the interaction between the CoP members in the form of collaboration, support, and knowledge sharing; and (3) Practice, a shared set of experiences, knowledge, and approaches the CoP creates. According to Wenger et al. [35], CoPs can be distinguished from other group structures, like formal departments, operational teams, project teams, and informal networks, based on their purpose, members, boundaries, motivation, and lifespan. The authors argue that CoPs are intended *"to create, expand, and exchange knowledge, and to develop individual capabilities."* Moreover, people join the community based on their *"expertise or passion for a topic"*, its boundaries are fuzzy, and members stay together due to *"passion, commitment, and identification with the group and its expertise."* Finally, a CoP *"evolves and ends organically"*, if no longer needed. In contrast, other group structures differ in at least one of the mentioned characteristics.

Multiple researchers found that CoPs can vary in different aspects. For example, Wenger et al. [35] state that communities can differ in their *scope*, which can be within a business unit, across business units, and organizational boundaries [35]. Moreover, how *official* the CoPs are can vary from unrecognized to institutionalized, influencing the degree of organizational steering and support [35]. According to Wenger et al. [35], CoPs also differ in their *size*, *lifetime*, *member distribution*, *homogeneity*, and *initiation* (top-down vs. bottom-up) [35]. Jassbi et al. [9] extend these aspects, e.g., by *member selection* (open vs. closed), *enrollment* (voluntary vs. mandatory), and *organizational support* (high vs. low).

According to Wenger et al. [35], CoPs can have short- and long-term values for their members and the organization. Members can get help with challenges, access to expertise, feel a sense of belonging, develop personally, and create a network [35]. Likewise, Fontaine and Millen [7] report a potential to improve the working environment, job satisfaction, and collaboration. Organizations can benefit from problem-solving, time savings, knowledge sharing, identified synergies across units, and reused resources [35]. In the long term, CoPs can positively influence innovation, strategic capabilities, or new strategies [35]. According to Lesser and Storck [15] and Fontaine and Millen [7], the communities can help respond faster and improve operational efficiency.

Scaled agile frameworks, such as SAFe [24], Spotify [12], LeSS [14], or Disciplined Agile [5] recommend organizations applying agile methods at scale to establish CoPs. According to the frameworks, the communities are relevant to connect experts with specific capabilities, typically distributed over cross-functional teams, to allow them to exchange and evolve shared practices [5,24]. CoPs can help foster self-organization [14], continuous learning [5,24], and leverage all experiences and knowledge existing within an organization [24].

Multiple researchers have studied CoPs in large-scale agile software development. Several of these authors investigated individual communities in-depth. For example, Silva and Doss [28] report on their experience establishing a company-wide Agile Coach CoP, which helped encourage learning, mutual support, and agile methods. Sporsem et al. [31] investigated a CoP that supported internal start-ups through experience and knowledge sharing, problem-solving, and the creation of shared experiences. Detofeno et al. [1] report on the establishment of a CoP within an organization that successfully contributed to technical debt management. Moe et al. [17] and Paasivaara and Lassenius [22] studied a corporate-level CoP at Ericsson, focusing on their decision-making power. Moe et al. [17] conclude that CoPs can serve as a bottom-up alignment mechanism between teams and empower them to make decisions.

Other authors investigated the establishment of CoPs within a single organization. Kähkönen [10] studied different CoPs within Nokia and emphasized their importance to succeed in agile multi-team settings. Korbel [13] reports on his experience of establishing different CoPs within an agile program to align distributed teams and individuals and foster continuous improvement. The teams were highly involved in the creation, proposing the CoPs' themes such as Scrum Mastering, DevOps, and UX. A few researchers provided an overview of what different CoPs they found within the organization they investigated. For example, Šmite et al. [30] investigated corporate-level communities at Ericsson, implemented to support the agile transformation and promote a participative culture.

The CoPs, consisting of unit representatives, were responsible for aligning all distributed units, promoting autonomy, and sharing knowledge. According to the authors, the CoPs created value for the organization, its units, and its members. The authors analyzed the CoPs regarding their mission and scope, authority, membership characteristics, scope, and meetings. Šmite et al. [29] studied multiple CoPs at Spotify (called "Guilds"), intended to improve decision-making, enhance support, and promote collaboration and cross-organizational knowledge sharing. The authors analyzed aspects like community members, scope, distribution, meetings, size, values, challenges, and focus (e.g., learning, onboarding and support, alignment, strategy development). Moreover, Paasivaara and Lassenius [20,21] studied CoPs at a unit of Ericsson with members located at multiple sites. The authors identified knowledge sharing and learning, coordination, technical work, and organizational development as purposes of the studied CoPs.

Uludağ et al. [32] identified Architecture CoPs applying collaborative decision-making in two organizations as part of their research on enterprise architecture in scaled agile contexts. Finally, Monte et al. [18] take an overarching view by investigating how CoPs are used to support agile teams through a literature review. The authors identified knowledge sharing, coordination, decentralized decision making, and fostering agile at scale as applications.

Almost all mentioned research endeavors studied individual or multiple CoPs within a single organization, hereby providing initial insights on why and what CoPs are established in large-scale agile software development. Still, no broad empirical overview across organizations exists.

3 Methodology

Study Design. We conducted semi-structured expert interviews to answer our RQs [8,19,27]. To ensure rigorous data collection, we respected the guidelines of Myers and Newman [19]. The chosen research design relies on qualitative data collection and is suitable as establishing CoPs in large-scale agile contexts is a problem in practice [27]. This study combines exploratory, descriptive, and explanatory elements. Table 1 presents an overview of the study participants. In two rounds, we interviewed 39 experts from 18 organizations. We combined convenience and purposive sampling [11]. Potential interview partners were contacted directly, e.g., via e-mail, and we shared our research endeavor with existing contacts and networks. Still, the selection was intentional to interview experts who work in scaled agile organizations or contexts [4], having CoPs established. We only interviewed experts involved in the communities as leads, members, or stakeholders (e.g., managers supporting CoPs). Some experts are involved in multiple CoPs in different roles. The interviewees have various job roles and industry backgrounds, leading to a *"variety of voices"* covering multiple viewpoints [19].

Data Collection. We conducted the first interview round between February and May 2023 (23 experts, 13 organizations) and the second (16 experts, five organizations) between November 2023 and January 2024. The majority of interviews

Table 1. Interview partners

Expert	Organization	Role	CoP Role	Experience (Years)*
E1	SoftwareCo1	Manager	Lead, Stakeholder	11–15
E2	InsureCo1	Enterprise Architect	Member	1–2
E3	SoftwareCo2	Agile Coach, Program Manager	Lead	11–15
E4	ConsultCo1	Manager	Lead	6–10
E5	SoftwareCo2	Software Architect	Member	16–20
E6	ConsultProj	Consultant, Q&A Specialist	Member	1–2
E7	CarCo1	Manager, Agile Coach, CoP Lead	Lead	6–10
E8	SoftwareCo2	Security & Infrastructure Expert, Scrum Master	Lead, Member	3–5
E9	SoftwareCo2	Developer, Scrum Master	Member	11–15
E10	CarCo2	Agile Coach	Lead, Member	3–5
E11	ConsultCo1	Business Analyst	Lead	6–10
E12	SoftwareCo2	Scrum Master	Lead	3–5
E13	ElectroCo	Agile Coach, Manager	Member	6–10
E14	ElectroCo	Agile Coach	Lead	1–2
E15	FoodCo	Agile Coach	Lead, Member	6–10
E16	SoftwareCo2	Scrum Master	Lead	6–10
E17	ConsultCo2	Agile Coach, Consultant, Product Owner (PO)	Lead	11–15
E18	ConsultCo1	Agile Coach, Scrum Master	Lead	6–10
E19	ConsultCo3	Consultant	Lead	16–20
E20	TeleCo1	Developer, Agile Coach	Lead	11–15
E21	InsureCo1	CoP Lead	Lead	3–5
E22	HealthCo	Software Architect	Member	11–15
E23	InsureCo1	Agile Coach, Enterprise Architect	Lead	6–10
E24	FashionCo	Enterprise Architect	Lead	3–5
E25	TransportCo	Solution Architect	Member	1–2
E26	TransportCo	Solution Architect	Lead, Member	1–2
E27	RetailCo	Manager	Lead, Member	16–20
E28	TransportCo	System Architect	Lead, Member	3–5
E29	TransportCo	Enterprise Architect, Manager	Member	6–10
E30	RetailCo	Project Manager	Lead	1–2
E31	TeleCo2	Manager	Stakeholder	11–15
E32	TransportCo	PO	Lead	6–10
E33	TeleCo2	Organizational Developer	Stakeholder	6–10
E34	TeleCo2	Enterprise Architect	Member	6–10
E35	InsureCo2	Enterprise Architect	Lead, Member	6–10
E36	TeleCo2	Agile Master	Member	3–5
E37	TeleCo2	Organizational Developer	Lead, Member	1–2
E38	TeleCo2	Disciplinary Leader	Member	11–15
E39	TeleCo2	Agile Master	Lead	6–10

had a duration of 40–60 minutes. Each interview started with an introduction to ensure a shared understanding of relevant concepts. All interviews had the same outline, with slight changes after the first round, to improve the data collection based on the experience gained and to collect data about challenges and best practices. We divided the interviews into three consecutive phases: (1) questions about the interviewees' backgrounds (e.g., experience in large-scale agile settings*), (2) questions about CoPs within their organization, (3) questions on how research can support practice in establishing CoPs in the first round and regarding challenges in the second round. The questions of the last two sections were open, allowing our interview partners to share their thoughts in detail. We conducted all interviews using videoconferencing tools. All interviews, besides one, were recorded and subsequently transcribed. Besides the interviews, we included relevant data sources (e.g., websites) shared by the interviewees to facilitate the data triangulation.

Data Analysis. The results presented are a subset of the collected data. We plan to communicate our other insights in later publications. We coded and analyzed the collected data based on the guidelines by Miles et al. [16] and Saldaña [25], with a two-cycle approach. First, we used inductive coding and summarized meaningful data fragments with descriptive codes. Then, these codes were assigned to second-cycle codes of two abstraction levels: (1) a set of high-level codes developed deductively based on the interview design and relevant concepts (2) lower-level codes developed inductively while analyzing the first-cycle codes, reflecting discovered concepts and patterns. In case of uncertainties or missing information, we contacted the interviewees. We included all group structures fulfilling the CoP definition [35], independent of their naming.

4 Results

In the following, we present the results of our interview study.

4.1 Context

Table 2 provides an overview of the experts' development organizations, which do not necessarily equal the whole IT department of the experts' companies.

4.2 Reasons for Establishing CoPs

We identified various reasons for establishing CoPs, summarized in Table 3, sorted by frequency. In the following, we present the most common reasons.

Promoting Knowledge Sharing. All CoPs we studied were established to foster knowledge sharing regarding different topics or a specific craft. The communities are intended to break down knowledge silos and allow benefiting from the knowledge and experience of others. For example, E15 explains that the Agile Coach CoP within FoodCo was founded as the Agile Coaches *"need some way of distributing experiences, [so] not everybody needs to learn from the same mistakes by themselves. But we want to create a shared pool of knowledge."*

Promoting Collaboration. Another common reason for establishing communities is to promote collaboration between employees and different organizational areas. E18 highlights that the communities are *"not only [for] exchanging ideas but really getting support from others and working on topics together."* The members support each other, build knowledge, and collaborate to create results and

Table 2. Development organization of the interview partners

Organization	Description
SoftwareCo1	20–30 employees split into two teams develop one software solution. E1 works with both teams. The company applies no specific scaling agile framework.
InsureCo1	E2, E21, and E23 work for the IT department of the company's German part, which applies a custom scaling agile framework containing elements of Spotify, LeSS, and SAFe. More than 2000 employees are working in 20 tribes, each having 2–25 teams.
SoftwareCo2	The company employs more than 30.000 developers. E9, E12, and E16 work on commerce, E5 and E8 on manufacturing software products. E3 works in software delivery. The areas we investigated combine elements of different frameworks.
ConsultCo1	The development part of ConsultCo1 comprises more than 6000 employees. The applied scaling agile framework is project-specific, often based on LeSS. E4, E11, and E18 all work on the same customer project: 450 employees divided into 35 feature teams grouped in eight programs are responsible for 40 applications.
ConsultProj	E6 is a consultant supporting a project of a company with more than 100 employees, including seven Scrum teams, combining elements of multiple frameworks.
CarCo1	E7 is working in a particular area of the company's enterprise IT, which has around 600 internal employees and consists of several domains. Within the company's overall IT, the scaled agile framework depends on the area, e.g., SAFe, LeSS, or custom.
CarCo2	E10 is working in the company's enterprise IT, consisting of around 900 internal employees and externals, divided into 120 to 140 different teams. The applied scaled agile framework is custom, oriented on SAFe.
ElectroCo	The company's IT department comprises more than 800 employees in more than 100 product teams, grouped into group-domains and sub-domains. E13 and E14 are involved in several domains. The organization is organized inspired by Spotify.
FoodCo	Over 2000 employees work in the company's IT organization, organized in 150–200 teams in different domains. The applied framework is custom, based on LeSS. E15 works on the IT organizational level.
ConsultCo2	ConsultCo2 is a sub-company of a large consulting firm. The sub-company consists of multiple hundred employees. E17 is part of a team of around 50 consultants. The applied scaling agile frameworks depend on the client.
ConsultCo3	E19 is an independent consultant, leading and organizing a cross-organizational CoP with members of multiple organizations.
HealthCo	The company's IT organization has over 1000 employees, organized in projects. E22's project involves around 250 developers divided into 30 teams. The applied scaling agile framework is custom, containing elements of SAFe.
TeleCo1	E20's department has its own framework combining elements of LeSS and Scrum-of-Scrums. One hundred developers in eight teams develop a software solution.
FashionCo	The company's IT department has around 400–500 employees, working in different domains. Some parts of the IT work agile, applying a custom framework inspired by SAFe. E24 works independently of any agile team structure.
TransportCo	The IT applies SAFe, with overall more than 1300 employees. E25, E26, and E28 work in different ARTs and large solutions, E29 and E32 work independently. TransportCo is supported by an IT sub-company without a specific framework.
RetailCo	The company's IT has over 2500 employees in different departments. E27's department has around 230 employees and 30 product teams. Twenty so-called streams cover cross-departmental projects. The IT organization applies its own framework. E30 works on the overall IT department level.
TeleCo2	The IT department is part of the technology area and has around 1000 employees. The department uses a custom framework, with naming based on Spotify. The 10–15 tribes consist of in total 30 teams. All interviewees work either with multiple teams or on the overall IT department level. E37 organizes a cross-company CoP.
InsureCo2	Within the agile part of the company's IT more than 220 employees work in four value streams of 4–8 teams, following a customized framework. In addition, the IT has multiple projects running. E35 works on the overall IT department level

solutions. For example, within ConsultProj's Scrum Master CoP, the members collaboratively develop *"common values, [...] plan suggestions, improvements that might be a bit more sound than if you just asked single individuals"* (E6).

Fostering People Development. Many organizations establish CoPs to foster the development of their employees. CoP members can learn and improve their skills together, and new employees can benefit from the knowledge of experienced members. E11 highlights that with CoPs *"people can learn faster, [...] become better in their job because they know the right people to talk to, [...] to learn from."*

Improving Efficiency. Another motive for establishing CoPs is to improve efficiency. The communities can help identify synergies, solve common challenges, and reuse, e.g., solutions. At SoftwareCo2, for instance, a technology-specific CoP helps developers to be aware of what *"other teams are doing with this technology so that whether we can reuse it or we can benefit from this"* (E5).

Fostering Networking. Another purpose of CoPs is to promote networking, as they can help their members meet people interested in a specific topic or with a particular role working in other areas. For example, E16 explains that the Scrum Master CoP within SoftwareCo2's commerce area helps *"to meet each other and to know what kind of Scrum Masters [...] we have in our organization."*

Distributing Information. Another common reason for establishing CoPs is distributing information to a target group. E14 highlights that the communities can help *"to get information from other areas [...] so that you can also create a bigger picture, a clearer picture about the things which are going on in the company."* Within SoftwareCo2, e.g., a CoP for Scrum Masters in commerce helps communicate information relevant to all employees in this role. According to E16, the *"CoP is a very crucial [...] place where we can communicate changes."*

Fostering Empowerment. Many CoPs are established to empower employees to act self-organized, take responsibility, make decisions, and influence actively. For example, E22 argues that the communities help *"avoiding discussions between people who are not really involved in the topics [...], have to feel the pain about their decision."* According to E11, communities are empowering employees as they *"give them the platform also to involve themselves."*

Aligning Across the Organization. Within several organizations, CoPs were established to align teams, areas, and specific roles across the organization. For example, for the ConsultProj, communities are needed *"to get people aligned"* as the project is *"very large"*, spanning *"different geographical areas, [..] time zones, [...] [and] various companies"* (E6). Likewise, E7 highlights that *"people with a certain task, [...] should align horizontally over the organization."*

Supporting the Agile Transformation. Some CoPs we investigated were established to support transitioning to agile. The communities can help to spread and create knowledge and experience regarding agile practices. Moreover, E29 highlights CoPs themselves *"bring agility into the organization."* Within SoftwareCo2, e.g., a central Scrum Master CoP was formed as *"there was the need*

Table 3. Reasons for establishing CoPs

Reason	Organization(s)	Expert(s)
Promoting knowledge sharing	All organizations	E1–12, E14–33, E35–39
Promoting collaboration	CarCo1&2, ConsultCo1–3, ConsultProj, ElectroCo, FashionCo, FoodCo, HealthCo, InsureCo1&2, RetailCo, SoftwareCo1&2, TeleCo2, TransportCo	E1, E2, E4–11, E13, E15, E17–19, E21–26, E28–31, E34, E35–39
Fostering people development	CarCo1, ConsultCo1, ElectroCo, FashionCo, FoodCo, HealthCo, InsureCo1&2, RetailCo, SoftwareCo1&2, TeleCo1&2, TransportCo	E1, E2, E4, E7, E8, E11, E13–15, E20–24, E27–29, E31, E33, E36, E38
Improving efficiency	CarCo1, ConsultCo1&2, ConsultProj, FoodCo, HealthCo, InsureCo1, RetailCo, SoftwareCo1&2, TeleCo1&2, TransportCo	E1, E2, E4–8, E11, E12, E15–18, E20–E23, E25–28, E30–32, E34, E36, E37, E39
Fostering networking	CarCo1, ConsultCo1–3, ElectroCo, FashionCo, FoodCo, InsureCo1&2, RetailCo, SoftwareCo2, TransportCo	E2–5, E7, E11, E13–19, E23–30, E35
Distributing information	CarCo1, ConsultCo1–3, ConsultProj, ElectroCo, InsureCo1, SoftwareCo2, TeleCo2, TransportCo	E2, E5–9, E11, E14, E16–19, E23, E25, E26, E28, E37
Fostering empowerment	ConsultCo1&2, ElectroCo, FashionCo, FoodCo, HealthCo, InsureCo1, RetailCo, TeleCo2, TransportCo	E2, E11, E13, E15, E17, E19, E22–25, E27–29, E32, E34, E38
Aligning across the organization	CarCo1, ConsultCo1, ConsultProj, ElectroCo, InsureCo1&2, RetailCo, SoftwareCo2, TeleCo2, TransportCo	E2, E5–7, E11, E13, E16, E23, E25–28, E30, E34, E35
Supporting agile transformation	CarCo1&2, ConsultCo1, FashionCo, HealthCo, InsureCo1, RetailCo, SoftwareCo2, TeleCo2, TransportCo	E2, E3, E7, E8, E10, E11, E22, E24, E27–29, E37, E39
Improving products and services	ConsultCo1&2, FoodCo, InsureCo2, RetailCo, SoftwareCo1&2, TeleCo2, TransportCo	E1, E5, E9, E11, E15, E17, E18, E25, E34, E35
Coordinating organizational units	CarCo1, ConsultProj, FoodCo, RetailCo, TransportCo	E6, E7, E15, E27, E28, E30, E32
Establishing a safe environment	ConsultCo1&3, ElectroCo, FoodCo, TeleCo2	E11, E14, E15, E19, E36, E38
Integrating new topics and trends	CarCo2, SoftwareCo1, TransportCo	E1, E10, E28

then to offer a capability to exchange among each other because Scrum master is really a special role compared to a traditional project manager" (E3).

4.3 Characteristics of the Established CoPs

In the following, we present the characteristics of the CoPs we investigated, without considering how they are organized internally (e.g., meeting frequency).

Initiation. In most organizations we investigated, CoPs initiated top-down by a management decision, and CoPs created bottom-up by employees exist. In a few cases, we only found communities initiated bottom-up (CarCo2, TeleCo1) or top-down (ConsultCo1, ConsultProj, HealthCo). The driver for initiating CoPs can change over time. Initially, in FoodCo, communities for agile topics and roles were started by individuals who saw a need. Then, management recognized the value of CoPs and initiated them for non-agile themes like security. In

ConsultCo2 and ElectroCo, management initiated the first CoPs, and employees later followed. Overall, slightly more than half of the CoPs we found were top-down-initiated. Most communities created to empower employees, establish safety, and improve offerings were initiated top-down. CoPs for new trends were mainly initiated bottom-up. Employees of one participating firm initiated the cross-company CoPs.

Target Group. Most organizations we studied have CoPs for specific roles (role-based) and independent of any role (topic-based). A few organizations have mainly one type (e.g., SoftwareCo1, ConsultCo1, ConsultProj, TeleCo1&2). Role-based CoPs are often open to others, e.g., for interested employees.

Themes. We identified various themes for which CoPs are established (see Table 4). The interviewees reported at least one community around *Agility* for almost every organization. Most of these communities are role-based, e.g., for Scrum Masters. Also, the cross-company CoPs are about agility. Another common theme is *Architecture.* Most Architecture CoPs are role-based. Many organizations also have communities for *Software Development*, some focusing on a particular technology or method, such as Citizen Development (TransportCo). Moreover, several organizations have, mainly top-down initiated, communities for classical and agile *Management and Leadership* roles (e.g., Disciplinary Leadership CoP (CarCo1, TeleCo2)) and *Security.* Most CoPs around *Product Management* are role-based PO CoPs. In addition, several organizations have mainly top-down initiated communities dealing with *Testing*, *UI/UX*, and *Quality*, e.g., software quality. Some organizations established, mainly topic-based, CoPs for *Cloud* and *Transformation and Change.* The latter include, e.g., cultural (CarCo1) and agile transformation (FashionCo). Finally, some communities for *Business Analysis* and *Digitialization and Innovation* exist, for example, for digitalizing transportation at TransportCo. Next to the presented themes, we found others implemented by only a few organizations, e.g., *AI*, or *DevOps.*

Openness. In many expert organizations, CoPs are open to everyone. Still, in multiple organizations, at least a few closed communities exist (ConsultProj, ConsultCo2, FashionCo, FoodCo, InsureCo1&2, RetailCo, SoftwareCo2, TeleCo2, TransportCo). These closed communities are mostly role-based or, if topic-based, initiated top-down. One studied cross-company CoP is closed.

Organizational Acceptance. Top-down-initiated CoPs are, by nature, accepted by management. Also, the bottom-up established communities we investigated are tolerated by the management, either due to official approval or as employees generally have the freedom to establish them. A few CoPs in some companies are even led by managers (e.g., ConsultCo1, ConsultProj, RetailCo, SoftwareCo1&2, TeleCo2). For example, the PO CoP within RetailCo is led by the Head of Product. We only found two CoPs of which management is unaware: a local Scrum Master CoP at SoftwareCo2 and the cross-company CoP of E37. Still, the companies' culture gives employees enough freedom to engage in such

Table 4. CoP themes

Theme	Organization(s)
Agility	CarCo1&2, ConsultCo1–3, ConsultProj, ElectroCo, FashionCo, FoodCo, HealthCo, InsureCo1, RetailCo, SoftwareCo2, TeleCo1&2, TransportCo
Architecture	CarCo1&2, ConsultCo1, ElectroCo, FashionCo, FoodCo, HealthCo, InsureCo1&2, RetailCo, SoftwareCo2, TeleCo2, TransportCo
Software Development	ConsultCo1, ElectroCo, FashionCo, FoodCo, HealthCo, InsureCo1&2, RetailCo, SoftwareCo1&2, TransportCo
Management and Leadership	CarCo1, ConsultCo1, ConsultProj, ElectroCo, FoodCo, InsureCo1, RetailCo, TeleCo2, TransportCo
Security	ConsultProj, FoodCo, InsureCo1&2, SoftwareCo2, TeleCo1, TransportCo
Product Management	FoodCo, HealthCo, InsureCo1, RetailCo, SoftwareCo2, TransportCo
Testing	CarCo1, ConsultProj, HealthCo, TeleCo2
UI/UX	ConsultCo1, InsureCo1, SoftwareCo2, TransportCo
Quality	HealthCo, InsureCo1, SoftwareCo1&2
Cloud	ConsultProj, FoodCo, InsureCo1
Transformation and Change	CarCo1, ConsultCo2, FashionCo
Digitalization and Innovation	ConsultCo2, RetailCo, TransportCo
Business Analysis	ConsultCo1, InsureCo2, TeleCo2

an exchange. In ConsultCo2 and FoodCo, establishing a CoP is formalized. For example, FoodCo has a *"form which needs to be filled [...] to create a CoP"* (E15).

Decision-Making Power. In some organizations, all CoPs we investigated can either make decisions affecting their area of expertise or influence them (CarCo1, ConsultCo1&2, ConsultProj, FoodCo, HealthCo, InsureCo1&2, SoftwareCo1, TeleCo1&2). In many other organizations, the decision-making power depends on the community (CarCo2, FashionCo, RetailCo, SoftwareCo2, TransportCo). Overall, most communities we studied have decision-making power or can influence decisions. CoPs established to foster empowerment largely have the power to at least influence decisions. The communities without the power to make decisions were mainly initiated bottom-up. Community decisions are often agreements, best practices, guidelines, or standards. For example, CoPs within ConsultCo1 are *"working on a guideline on how certain roles should behave in a project. [...] It's not forcing any projects to apply this, but it's definitely something that will [have] an influence"* (E18). The cross-company CoPs we studied are not entitled to make any decisions affecting the involved organizations.

Organizational Support. While the time employees spend within CoPs is funded, most communities we studied have no dedicated budget. Only within a few organizations (some) CoPs have a budget or sponsor funding, e.g., external guest speakers (CarCo2, FoodCo, InsureCo1, RetailCo, TeleCo1, TransportCo). A minority of CoPs have full-time leads. Often, Agile Coaches or a team support (ConsultProj, ElectroCo, FoodCo, HealthCo, InsureCo1, RetailCo, TeleCo2) and management promotes (some) CoPs actively (ConsultProj, ConsultCo2, FashionCo, HealthCo, InsureCo1, RetailCo, SoftwareCo1&2, TeleCo2, TransportCo).

Steering. In many organizations, at least some CoPs we investigated are monitored and steered by management to a certain degree, e.g., regarding content and strategy (CarCo1&2, ConsultCo1&2, ConsultProj, FashionCo, FoodCo, InsureCo1, RetailCo, SoftwareCo1&2, TeleCo2, TransportCo). More top-down than bottom-up initiated communities belong to this group.

Participation. In most experts' organizations, participation in CoPs is voluntary or sometimes expected. Some companies additionally have a few mandatory communities (CarCo2, FashionCo, InsureCo1, SoftwareCo2). These CoPs are officially approved or initiated, and often steered by management. Also, these CoPs mostly have access to funding and decision-making power. The cross-company CoPs are voluntary.

Scope. Within some organizations we only found CoPs spanning across the whole development organization (ConsultCo2, ConsultProj, TeleCo1&2, SoftwareCo1). These development organizations often have no "team of teams" structure grouping teams, e.g., into ARTs. Other organizations with "teams of teams" or even more complex structures mostly have communities on additional levels. Some organizations have CoPs restricted to a single or a few "team of teams" (CarCo1, FashionCo, FoodCo, HealthCo, InsureCo1&2, RetailCo, SoftwareCo2, TransportCo), for representatives of each "team of teams", or for roles with leadership responsibility, e.g., for a "team of teams," mainly with decision-making power (ConsultCo1, ElectroCo, FoodCo, InsureCo1, RetailCo, SoftwareCo2, TransportCo). For example, InsureCo1 has a community for its Tribe Leads. The most common scope of CoPs we found is the overall development organization. Especially topic-based communities tend to have a broad scope. Moreover, some companies (e.g., CarCo2, TransportCo) have CoPs covering the whole organization, not just IT. The cross-company CoPs go beyond a single organization. Almost all CoPs we investigated have members from different sites in Germany or other countries. The only single-site communities we found were all communities of TeleCo1 and InsureCo2, and two Scrum Master CoPs and a PO CoP at SoftwareCo2.

Size. The size of the CoPs we investigated ranges from less than ten members to several hundred. Naturally, the organization's size plays a role. For example, CoPs within SoftwareCo1, a small company, could not reach hundreds of members. The topic-based and open communities we studied are often bigger than role-based and closed ones. Likewise, the CoPs with a broader scope are in many cases bigger than ones, e.g., for a single "team of teams". The CoPs we found intended to be a safe space for employees tend to have less than 50 members.

5 Discussion

In this section, we discuss our key findings and the limitations of our study.

5.1 Key Findings

Our study identified promoting knowledge sharing as the most common reason for establishing CoPs. Likewise, Paasivaara and Lassenius [21] and Šmite et al. [30] found knowledge sharing as at least an implicit goal behind all communities in the scaled agile organizations they studied. As in both studies, the CoPs we investigated have various purposes. The most common reasons are to enhance collaboration, people development, efficiency, and networking. These findings align with the values CoPs can provide according to Wenger et al. [35] and the results of other authors studying communities in scaled agile settings [17, 21,28–31]. Moreover, we identified promoting empowerment, alignment and the agile transformation as common motives. Likewise, Moe et al. [17] found that CoPs can empower through community-based decision-making. Other authors report on CoPs intended to align distributed units and teams within scaled agile settings [13,29,30] and conclude that CoPs can support the agile transformation [18,21,30].

The communities we studied cover a variety of themes not specific to agile methods. Besides CoPs for agile roles or about agility, we identified numerous ones for themes like architecture or security. Likewise, other related studies mainly found CoPs for themes not specific to agile practices [21,29,30]. While both topic- and role-based communities were common in our study, some themes (e.g., architecture) are mainly addressed in role-based CoPs, and others (e.g., transformation and change) are the subject of topic-based CoPs.

Our study shows that in most organizations, CoPs are initiated top-down and bottom-up. Communities for several themes, like quality, were primarily initiated by managers. The same applies to CoPs created for specific reasons, like empowerment. CoPs focusing on new trends were mainly created bottom-up. We also found that the drivers for CoPs can change over time, e.g., if management or individuals start recognizing their value. Likewise, Paasivaara and Lassenius [21] report a change in the initiation from managers to those who needed them.

Most communities we investigated are open and voluntary. In line with our findings, other studies found mainly open [29] and voluntary communities [21, 30]. According to several authors [21,35], openness is crucial. Moreover, Wenger et al. [35] see mandatory participation as a weakness, whereas Šmite et al. [30] found that it can ensure all organizational areas are represented.

The CoPs in our study are accepted by management due to top-down initiation, approval, or the overall organizational culture. While the time spent in communities is funded, most lack a dedicated budget, regardless of whether they are initiated top-down. Similarly, in other studies [21,29,30], CoPs are officially recognized. Only Šmite et al. [29] report a dedicated CoP budget. Most CoPs we studied, particularly if established to foster empowerment, can make or influence decisions in their areas of expertise. Several authors (e.g., [21,30]) investigated CoPs entitled to make decisions or give recommendations and concluded that this authority is needed. Our study also showed that management often steers communities, particularly if initiated top-down. Likewise, Wenger et al. [35] connect institutionalization with potential overmanagement.

In our study, CoPs most often span the whole development organization, especially if topic-based. In addition, most organizations with a “team of teams” structure have CoPs on more levels. The scopes we found align with the categories Wenger et al. [35] and the scopes in agile organizations other researchers describe [21,29,30]. As in these studies, the CoPs we studied mainly cover different sites. The size of the communities we investigated ranges from less than 10 to several hundred members. Topic-based and open CoPs tend to have more members than role-based and closed ones, potentially due to a broader target group. Same applies to CoPs spanning the whole development organization. A similar pattern can be found when comparing studies of CoPs with different scopes ([29] vs. [30]).

5.2 Limitations

We used the assessment schema of Runeson and Höst [23] to evaluate threats to our study’s validity. We addressed the threat of construct validity by clarifying relevant concepts and potential ambiguities during each interview. To overcome the threat of external validity, also influenced by our sampling strategy, we interviewed 39 experts with different roles from 18 organizations in different industries to increase generalizability. For achieving reliability, two researchers were involved in the study process, and we followed guidelines for the data collection and analysis [16,19,25]. Whenever the interpretation of data needed clarification, we contacted the interviewees. We tried to address threats to internal validity by following the same outline to collect data and building on patterns identified between the organizations and communities we studied. Still, quantitative research is needed to validate the causal implications.

6 Conclusion and Future Work

Organizations applying agile methods at scale have to cope with increased complexity and challenges like redundant work, silo thinking, lacking collaboration, and inconsistent agile practices [2,3]. Existing research provides first insights into how CoPs can address these issues and support organizations applying agile methods at scale [21]. However, a broad overview of why large agile organizations establish CoPs and what these communities look like is missing. To fill this gap, we conducted an expert interview study comprising 39 experts from 18 organizations. We found that next to benefiting from the known advantages of CoPs, organizations applying agile methods at scale establish them to foster empowerment, strengthen alignment, and drive their agile transformation. Moreover, CoPs focus not only on agile but also on classical themes such as architecture. Most communities are open and voluntary but not necessarily established bottom-up. Often, management initiates CoPs, e.g., to empower their employees, giving them the authority to make decisions in their area of expertise. CoPs are either explicitly or implicitly accepted by management, as the organizational culture provides the freedom for such, and often steered, especially if they are top-down initiated. Our study provides insights into CoPs in large-scale agile software development, which can inspire practitioners and allow future research. We aim to publish the remaining evidence on how communities are organized and identified barriers, challenges, and best practices. We also want to use quantitative methods to validate and extend our results. Moreover, building on our findings on why and what CoPs are established in scaled agile settings by enhancing them with additional empirical insights could allow for a well-founded taxonomy of CoPs in this context. Furthermore, we presented why organizations have established CoPs but did not provide insights into their success. Future research could, for example, focus on appropriate indicators or measurement approaches that allow organizations to evaluate the success and value of CoPs within scaled agile settings.

Funding Information. This research has been partially funded by the German Federal Ministry of Education and Research (BMBF) through grant 01IS23069.

References

1. Detofeno, T., Reinehr, S., Andreia, M.: Technical debt guild: when experience and engagement improve technical debt management. In: Proceedings of the Brazilian Symposium on Software Quality 2021, pp. 1–10. ACM, New York (2021)
2. Digital.ai: 16th Annual State of Agile Report (2022). https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf. Accessed 12 Apr 2024
3. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. J. Syst. Softw. **119**, 87–108 (2016)
4. Dingsøyr, T., Moe, N.B.: Towards principles of large-scale agile development. In: Dingsøyr, T., Moe, N.B., Tonelli, R., Counsell, S., Gencel, C., Peternsen, K. (eds.) International Conference on Agile Software Development 2014. LNBIP, vol. 199, pp. 1–8. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14358-3_1
5. Disciplined Agile: Communities of Practice (2024). https://www.pmi.org/disciplined-agile/people/communities-of-practice. Accessed 12 Apr 2024
6. Highsmith, J.A., Highsmith J.: Agile Software Development Ecosystems. Addison-Wesley Professional (2002)
7. Fontaine, M.A., Millen, D.R.: Understanding the Benefits and Impact of Communities of Practice. In: Knowledge Networks: Innovation Through Communities of Practice, pp. 1–13. IGI Global, Hershey (2004)
8. Fontana, A., Frey, J.H.: The interview: from structured questions to negotiated text. In: Handbook of Qualitative Research. 2n edn, pp. 645–672. SAGE, Thousand Oaks (2000)
9. Jassbi, A., Jassbi, J., Akhavan, P., Chu, M.-T., Piri, M.: An empirical investigation for alignment of communities of practice with organization using fuzzy Delphi panel. Vine **45**(3), 322–343 (2015)
10. Kähkönen, T.: Agile methods for large organizations-building communities of practice. In: Proceedings of the Agile Development Conference 2004, pp. 2–10. IEEE, New York (2004)

11. Kitchenham, B., Pfleeger, S.L.: Principles of survey research: part 5: populations and samples. ACM SIGSOFT Softw. Eng. Notes **27**(5), 17–20 (2002)
12. Kniberg, H., Ivarsson, A.: Scaling agile @Spotify (2012). https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf. Accessed 12 Apr 2024
13. Korbel, A.: Using Communities of Practice for Alignment and Continuous Improvement at DigitalGlobe. https://agilealliance.org/wp-content/uploads/2015/12/ExperienceReport.2014.Korbel.pdf. Accessed 12 Apr 2024
14. LeSS: Communities of Practice (2024). https://less.works/less/structure/communities. Accessed 12 Apr 2024
15. Lesser, E.L., Storck, J.: Communities of practice and organizational performance. IBM. Syst. J. **40**(4), 831–841 (2001)
16. Miles, M.B., Huberman, A.M., Saldaña, J.: Qualitative Data Analysis: A Methods Sourcebook, 4th edn. SAGE, Thousand Oaks (2019)
17. Moe, N.B., Šmite, D., Paasivaara, M., Lassenius, C.: Finding the sweet spot for organizational control and team autonomy in large-scale agile software development. Empir. Softw. Eng. **26**, 101 (2021)
18. Monte, I., Lins, L. Marinho, M.: Communities of practice in large-scale agile development: a systematic literature mapping. In: Proceedings of the Latin American Computer Conference 2022, pp.1–10. IEEE, New York (2022)
19. Myers, M.D., Newman, M.: The qualitative interview in is research: examining the craft. Inf. Organ. **17**(1), 2–26 (2007)
20. Paasivaara, M., Lassenius, C.: Deepening our understanding of communities of practice in large-scale agile development. In: 2014 Agile Conference, pp. 37–40. IEEE, New York (2014)
21. Paasivaara, M., Lassenius, C.: Communities of practice in a large distributed agile software development organization - Case Ericsson. Inf. Softw. Technol. **56**(12), 1556–1577 (2014)
22. Paasivaara, M., Lassenius, C.: Empower your agile organization: community-based decision making in large-scale agile development at Ericsson. IEEE Softw. **36**(2), 64–69 (2019)
23. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**, 131–164 (2009)
24. SAFe: Communities of Practice (2023). https://scaledagileframework.com/communities-of-practice/. Accessed 12 Apr 2024
25. Saldaña, J.: The Coding Manual for Qualitative Researchers, 4th edn. SAGE, Thousand Oaks (2021)
26. Schwaber, K., Beedle, M.: Agile software development with scrum. Series in agile software development. Prentice Hall (2002)
27. Seaman, C.B.: Qualitative methods in empirical studies of software engineering. IEEE Trans. Software Eng. **25**(4), 557–572 (1999)
28. Silva, K., Doss. C.: The growth of an agile coach community at a fortune 200 company. In: Agile 2007 (AGILE 2007), pp. 225-228. IEEE, New York (2007)
29. Šmite, D., Moe, N.B., Levinta, G., Floryan., M.: Spotify guilds: how to succeed with knowledge sharing in large-scale agile organizations. IEEE Softw. **36**(2), 51–57 (2019)
30. Šmite, D., Moe, N.B., Wigander, J., Esser, H.: Corporate-level communities at Ericsson: parallel organizational structure for fostering alignment for autonomy. In: Kruchten, P., Fraser, S., Coallier, F. (eds.) XP 2019. LNBIP, vol. 355, pp. 173–188. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19034-7_11

31. Sporsem, T., Tkalich, A., Moe, N.B., Mikalsen, M., Rygh, N.: Using guilds to foster internal startups in large organizations: a case study. In: Gregory, P., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming - Workshops 2021. LNBIP, vol. 426, pp. 135–144. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88583-0_13
32. Uludağ, Ö., Reiter, N., Matthes, F.: Improving the collaboration between enterprise architects and agile teams: a multiple-case study. In: Zimmermann, A., Schmidt, R., Jain, L.C. (eds.) Architecting the Digital Transformation. ISRL, vol. 188, pp. 347–366. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-49640-1_18
33. Uludağ, Ö., Philipp, P., Putta, A., Paasivaara, M., Lassenius, C., Matthes, F.: Revealing the state of the art of large-scale agile development research: a systematic mapping study. J. Syst. Softw. **194**, 111473 (2022)
34. Van Oosterhout, M., Waarts, E., van Hillegersberg, J.: Change factors requiring agility and implications for IT. Eur. J. Inf. Syst. **15**(2), 132–145 (2006)
35. Wenger, E., McDermott, R., Snyder, W.M.: Cultivating Communities of Practice: A Guide to Managing Knowledge. Harvard Business School Press, Boston (2002)

Slack Use in Large-Scale Agile Organizations: ESN Tools as Catalysts for Alignment?

Viktoria Stray[1,2](✉) and Astri Barbala[1]

[1] SINTEF Digital, 7034 Trondheim, Norway
[2] University of Oslo, 0373 Oslo, Norway
stray@ifi.uio.no

Abstract. This paper examines the role of the enterprise social networking (ESN) tool Slack in the daily work of software practitioners within NAV, a large-scale agile public sector organization. Based on 13 interviews with NAV developers, our case study explores how Slack is employed for knowledge sharing and daily communication across the organization. We used a newly developed framework for communication in agile teams as a theoretical lens. Through our analysis, we found that Slack use had become deeply integrated into the organizational culture and fostered alignment in three main ways: Promoting *communication transparency* through open discussions visible for developers organization-wide, enhancing *communication quality* with prompt responses and constant communication, and encouraging *communication discipline* through structured channels and threads. This study also unveiled some challenges, such as information overload and hindered focus. However, our findings suggest that if common hurdles are overcome, modern ESN tools can reshape how cross-organizational communication plays out in large-scale agile, reinforcing the agile principles of collaboration and motivated individuals.

Keywords: Collaboration Tools · Team Communication Platforms · Agile Software Development · Human and Social Aspects of Software Engineering · Online Collaborative Software · Instant Messaging

1 Introduction

The use of enterprise social networking (ESN) tools, such as Slack, has become increasingly popular as they enable agile software teams to collaborate efficiently [11,36]. ESN tools facilitate communication and networking within an organization and are specifically designed to support social interactions among employees, enabling them to share information, work together on projects, and build relationships [19]. Social interactions and networking is essential for both novice and mature agile teams when solving complex, unfamiliar, or interdependent tasks [34]. ESN tools are also called Online Collaborative Software [4], Team Communication Platforms [3], and workspace collaboration tools [4,17].

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 20–35, 2024.
https://doi.org/10.1007/978-3-031-61154-4_2

Agile methods, known for their emphasis on flexibility, adaptability, and team collaboration, have been widely adopted in software development projects, and in recent years in large-scale settings [15,16]. However, the specific dynamics of communication and collaboration in large-scale agile environments remain underexplored, particularly in the context of the public sector. Public sector organizations are characterized by complex bureaucracies and diverse stakeholder needs, and require efficient and effective communication tools to navigate their unique challenges. Further, the rate of innovation and the speed of development are often slower in the public sector compared to the private sector [21].

Previous research has shown that alignment and coordination between multiple teams in large-scale settings is a challenge [14,15]. Having too many dependencies to others has been reported as a top barrier for autonomous agile teams [26]. Yet, there is still a lack of studies looking into how these challenges play out in public sector organizations. Attempting to meet this research gap, this study highlights the challenges and benefits of using a leading ESN platform, Slack, in a large-scale agile setting in the public sector, pointing to how it can help organizations improve their communication and coordination strategies.

Slack was originally designed as a cloud-based platform for team communication, but has now moved beyond just a tool for sending messages. Research on Slack has shown that for many software teams the platform has turned into a workflow hub where they use bots, meet in ad-hoc 'huddles' and receive alerts from other tools [20]. Despite the growing use of Slack in agile software development (ASD) teams, however, there is a lack of research on how to optimize its use to improve communication and coordination, especially in a large-scale agile organizational context. This paper delves into the use of Slack within NAV, the Norwegian Labour and Welfare Administration, which can be classified as a large-scale public sector organization. NAV's IT department, counting more than 900 employees, has embraced Slack in their agile teams, not just as a tool for daily communication, but as a central coordination tool across the organization that supports their software development in a large-scale setting. Adopting an organizational perspective in order to grasp how Slack facilitates for or challenges alignment across the organization, we ask the following research question:

RQ: *How can the use of Slack facilitate or challenge organizational alignment in a public sector large-scale agile software development setting?*

In addressing our research question, we conducted fieldwork in NAV, which is an organization tasked with specific employment and welfare services. Crucial to Norway's welfare infrastructure, NAV administers around a third of the state budget and facilitates unemployment benefits, pensions, and provide social and financial security to the nation's citizens. The organization also employs thousands of caseworkers, which necessitates a comprehensive array of information systems designed to aid not only expert users within NAV, but also citizens, employers, healthcare providers, and family members. With its IT department being among the country's most extensive in software development, NAV handles vast and intricate data sets, essential for both individual citizens and societal stability. The organization also plays a critical role in producing vital public and official statistics that are instrumental in shaping national policy decisions.

By conducting 13 in-depth interviews with software developers in NAV, this research seeks to shed light on the integration of Slack into the organizational culture and its impact on the everyday life and practices of software developers. We aim to address the use of Slack and identify the challenges and benefits associated with its use. Through this exploration, the study contributes a deeper understanding of how digital communication tools are reshaping agile methods in the public sector, offering insights that could guide similar organizations in optimizing their communication strategies for better software development outcomes.

2 Background

2.1 Large-Scale Agile Development in Public Sector Organizations

Studies of agile software development in the public sector have tended to focus on themes such as scaling [18], project governance [23], and organizational implications of agile adoption [27]. What differentiates a public sector context from a private sector context will depend on the country in which the study is undertaken, for instance, the extent to which commercial enterprises are involved with public goods and governmental services such as law enforcement, public education, infrastructure, and public transportation. In the Norwegian context, non-profit digitalization initiatives dominate the public sector software development.

Still, national public sector organizations have often been at the forefront of implementing new methods and technologies such as both agile [15] and continuous software engineering (CSE) methodologies [6]. Data-driven initiatives have also been a focus in Norwegian public sector bodies [10], and the country has been regarded as a "global pioneer in incorporating new ideas about organizational architecture in software development" [7]. In investigating the changes taking place during NAV's digital transformation in the years between 2016–2020, Bernhardt [9] discovered that the key factors influencing the organization were the reconfiguration of its structure to facilitate team and product area creation, an overhauled sourcing strategy that favors bringing services in-house, a move to a contemporary application platform with a flexible architecture, and a shift from traditional to agile methodologies in product development.

In another study, Dingsøyr et al. [15] enquire into how public sector organizations often can be studied as "very large-scale agile development" contexts, highlighting the intricate interplay of numerous teams with expansive duties. Dingsøyr et al. observed that the organization they studied shifted from an initial large-scale agile method that blended agile practices with project management protocols to a more evolved form that integrates contemporary ideas in software development. This evolution led to the creation of independent teams, each specialized in a different product area. Dingsøyr et al. report that this change led to a dramatic boost in the frequency of product updates, escalating from biannual to daily releases. This shift also brought about a profound change in coordination strategies, empowering teams to develop their own methods of coordination, which resulted in a streamlined process with fewer middlemen and less inter-team reliance [15].

2.2 Communication in Agile Software Development

As Bablo et al. [5] notes, communication in agile teams is not just frequent but also informal and direct, enabling rapid responses to change and fostering a collaborative environment. For a long time, communication tools such as Slack have been vital in globally distributed teams, providing an asynchronous yet real-time communication medium supporting fast feedback [25].

Recently, there has been a shift in how agile teams work because of the pandemic that forced people to work from home. As a result, more and more people are working from home for several days per week, resulting in a hybrid way of collaboration [13]. The new collaborative working mode has affected the use of communication tools and coordination mechanisms. This was reported in a recent survey that found that the percentage of people who are collocated in the same office has changed from 51.9% to 16.9%, and 80% use instant messaging tools such as Slack and Viber [28]. Santos and Ralph [35] recently conducted a study on hybrid software teams that revealed that the absence of regular face-to-face communication may cause the teams to revert to ineffective communication mechanisms, and the teams should modify their communication methods to enable more frequent interactions.

Kostin and Strode [22] emphasize that communication is not merely a facilitator but a cornerstone of agile practices, and our study hence seeks to contribute to this literature. It is through efficient communication that agile teams can ensure transparency, manage evolving requirements, and maintain a continuous feedback loop with stakeholders. In their recent work, Kostin and Strode [22] propose a theoretical model for alignment in distributed agile teams. The model consists of four elements: alignment, communication transparency, communication quality, and communication discipline, where the latter three lead to alignment if successfully executed. *Communication transparency* refers to the openness in sharing all essential product-related information, ensuring all team members have access to the data they need for effective collaboration, *communication quality* emphasizes the effectiveness, clarity, and appropriateness of the information exchanged, ensuring it is understandable for everyone involved, and *communication discipline* involves adhering to established communication protocols and schedules, ensuring consistent and reliable information exchange to maintain effective collaboration and alignment with project goals. Lastly, *alignment* refers to the degree of which team members - and in our case; organizational peers - share a common understanding and commitment to project goals, methods, and values. Alignment is crucial for coordinated action, both in distributed teams, as in Kostin and Strode's case, but also in a large-scale agile organization where the risk of miscommunication is higher. Since the NAV developers we interviewed are all part of agile and often hybrid teams, we found this model to be a valuable lens for analyzing how they employed Slack in their everyday work.

2.3 ESN Tools

The role of ESN tools, such as Slack, extends beyond mere message exchange; they are integral in supporting the dynamic, iterative, and collaborative essence of agile methodologies. Slack was initially introduced as an alternative to traditional communication methods, such as emails and meetings, but has now become an essential tool in development environments due to its versatile and all-inclusive communication capabilities [20]. Slack was created by the American software company Slack Technologies and has been under Salesforce's ownership since 2020. It provides both freemium and premium subscription options, featuring capabilities like text messaging, sharing of files and media, voice and video calling, and group chat functionalities to facilitate team collaboration. It can be contended that Slack has significantly influenced how software development teams interact and work together [4,24,29]. Especially in agile software development, Slack can help overcome inter-team communication barriers [30], lower the threshold to ask other team members for help [38], decrease task allocation dependencies and increase awareness of what others are doing [37].

The use of Slack in a multidisciplinary academic team showed that its activity mirrored social interactions and project progress, correlating with important milestones and reflecting the team's multidisciplinary collaboration [4]. In another study, Calefato et al. [11] investigated the role of tool support in facilitating collaboration during agile development. Specifically, they focused on the implementation of a Slack workspace to enhance teamwork in agile environments. An analysis of Slack usage in engineering design teams revealed insights into team dynamics and communication patterns, such as the central role of leaders, the importance of emoticon reactions in communication, and the evolution of communication topics over time [2].

Although ESN tools facilitate synchronous communication, they also present several challenges. For instance, teams must figure out how to interact with one another inside the tool and how to balance the use of these tools with other communication forms, including meetings, emails, and phone calls [36]. It also happens that some people dominate conversations in Slack channels [32].

3 Methods and Study Design

Due to the exploratory nature of our study, we conducted an exploratory case study inspired by the approach of Runeson and Höst [31]. They outline five steps for conducting this type of study within software development research: (1) study design, (2) preparation for data collection, (3) data collection, (4) analysis of collected data, and (5) reporting. Following Walsham's [40] methodology for interpretive case study research in information systems, we acknowledge that case studies are not designed for statistical generalization. This can be considered a limitation of case studies and qualitative research broadly. Nonetheless, the study allows for analytical generalization because it presents a method for investigating the use of collaboration tools by software developers, and the challenges they face in this regard, which is a relevant issue for software teams globally.

The data collection conducted in this study was performed as part of a large research project looking into agile software development in the public sector. The data material consists of interviews with 13 developers in NAV as well as digital observations of a large Slack channel used by almost 200 of the developers in NAV's IT department. The company may be characterized as *large-scale*. Currently, NAV IT has around 900 employees encompassing more than 150 teams with developers, and operates with an annual budget of more than 100 million Euros. The employees include 30 architects, 80 designers, over 300 developers, 180 technicians (operations and infrastructure), as well as other leadership, advisory, and support positions. Each team is tasked with specific responsibilities, and the teams can select their preferred tools, technologies, and agile ways of working, thereby granting them a considerable level of autonomy. After the pandemic-related restrictions in mid-2021, the company gave people the flexibility to work from home, which many of the employees choose to do at least some days per week.

3.1 Data Collection

Of the 13 interviews we conducted, four interviews were performed face-to-face at NAV's head office and nine interviews were held via Teams. The interviews were conducted between November 7th and December 18th, 2023, and we gave the informants pseudonyms based on their interview number (Person1, Person2, etc., shortened to P1, P2 in the following). A semi-structured approach was followed, with each interview lasting 30–60 minutes, with an average of 47 min. In the interviews, we asked developers about their relationship with work tools, how they collaborate with colleagues and what contributes to them feeling satisfied at work. Further, we asked the informants to tell us about how they use Slack, detailing how often they use it, which channels they are part of, and what role Slack plays in their collaboration with their team and others in the NAV organization. Some also shared their screen and showed us their Slack interface, so we got an even better understanding of the channels they followed and which connections they communicated with.

In addition to interviews, we were also invited to join a NAV Slack channel in the organization. We regularly checked posts and discussions on this channel during the course of the data collection and analysis, and also used it as background material for our interviews, e.g., if one informant had shared or been part of a discussion on a specific topic. We paid attention to how posts were shared and by whom, how people started discussions under said posts, and how members used emoticon reactions in specific ways. The digital observation method, which allows for continuously following and analyzing a community's digital practices and interactions [8], was followed here.

3.2 Data Analysis

We used abductive thematic analysis to extract qualitative information through explicit codes, defined as patterns of meaning (themes) across all interview transcripts [39]. Abductive analysis enables researchers to apply established theories and concepts, as is typical in deductive analysis, while simultaneously uncovering fresh insights and perspectives directly from the data. This meant that we were constantly moving between relevant literature and data in our analysis process, allowing theory about the topic and our empirical material to mutually shape one another.

Through this process, we discovered the newly developed model for ASD communication [22], which we found to be a suitable analytical framework for our study to point at both the benefits and challenges of relying on Slack as a communication and coordination platform in a large-scale agile setting. After our initial coding process, we grouped the codes into interpretative sub-themes and subsequently into candidate themes that lend their tags from Kostin and Strode's model. The content and meaning of the three categories were, however, shaped by our data material, which regarded Slack use in a large-scale public sector organization rather than communication in globally distributed Scrum, as in the case of Kostin and Strode. However, we did not find this to be an obstacle during our analysis; on the contrary, it enriched the analytical process by making us thoroughly discuss each finding in accordance with the model.

4 Results

This section describes the findings from our investigation into the use of Slack as an ESN tool within NAV. Our analysis draws on interviews and digital observations. The results are described following the three categories that, according to Kostin and Strode [22], support alignment for agile teams: Communication transparency, Communication quality, and Communication discipline.

4.1 Communication Transparency

It was clear from our analysis that the widespread use of Slack's open channels was vital in fostering transparent communication between employees throughout the organization. This was especially important as many people in NAV were working from home. P5 stated: *"In our team, we mostly work from home and have one office day per week. Therefore, we use Slack a lot."*

NAV IT employees were encouraged by their peers to publish messages in open organization-wide Slack channels to ensure that everyone would have access to the information they were sharing. This practice ensured that all NAV's developers, regardless of their role or location, could be part of the conversation. Our digital observations revealed that not only the developers were using NAV's open Slack channels: Although Slack is mostly used by NAV IT, people from other departments that are members of cross-functional development teams also have access and use it in their teamwork or to answer questions, such as leaders and people from legal department. Thus, some section managers high up in the organization were also active members, providing answers to questions, taking part in discussions, and adding emoticon reactions to other members' posts. It was evident that there was an abundance of open channels within the NAV organization, such as for specific IDEs, front-end/back-end developers and universal design, which we also observed when interviewees showed us their Slack interfaces.

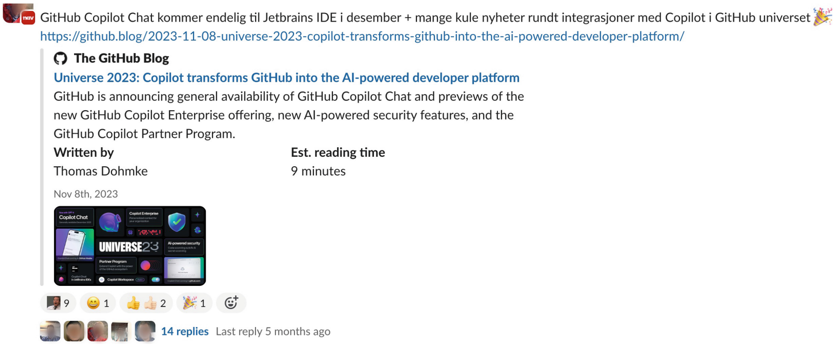
Several pointed to how the large-scale nature of the organization, meaning teams were spread out both in terms of location and product areas, made Slack's options for communication especially beneficial: *"I find that this organization greatly benefits from using Slack, as we have so many channels for so many things. I use Slack quite extensively when I have questions that we can't automatically answer within the team, or that no one in the team can respond to. We also use Slack within the team to some extent, and we use it to communicate with our external partners, or those we depend on, or who depend on us. So we use it for communication both within the team, across the organization, and externally."*, P3 said. P10 used Slack to keep updated on what other kinds of developers in the large-scale organization discussed, and said humorously: *"I think it is very valuable to have some insight into what those weird back-end developers are talking about".*

Other interviewees expressed that the transparency facilitated by Slack had not only enhanced their understanding of projects and tasks but also contributed to a more positive work environment. One developer, P8, went as far as saying Slack was so integral to his job as a NAV developer that he would consider quitting his position if the organization was to discontinue using Slack. All in all, there was a broad consensus among the developers we interviewed that the ability to easily share information and seek assistance had reduced barriers to communication, making it easier for individuals to contribute to discussions and decision-making processes. The preference for Slack over other tools like Microsoft Teams was noted by P5: *"Slack is easier for joining and leaving conversations, as well as for promoting open discussions, unlike Teams."*

However, the reliance on open channels on Slack also introduced challenges for communication transparency amongst the developers. For instance, P1 noted the tendency of some developers, especially new hires, to avoid public channels: *"Many find it scary to ask in open channels, so they ask me directly."* This comment points to a need for fostering a more open communication culture in the organization, that perhaps the architecture or affordances of Slack may not offer to a satisfying degree for such a large-scale organization. Another interviewee, P3, also showed concern that the less active users would not receive crucial information if all conversations and important messages were shared mainly on Slack. He said: *"I have noticed that some people remind other users that it's not certain we are all in the specific channel [they post information in], so using it as the main platform for information might not be good enough."*

4.2 Communication Quality

Slack's features, such as being able to edit messages after they are sent, and the ability to quickly share files or have conversations through a huddle - the platform's possibility for calling up people instantly in the same way as through



GitHub Copilot Chat kommer endelig til Jetbrains IDE i desember + mange kule nyheter rundt integrasjoner med Copilot i GitHub universet 🎉
https://github.blog/2023-11-08-universe-2023-copilot-transforms-github-into-the-ai-powered-developer-platform/

The GitHub Blog
Universe 2023: Copilot transforms GitHub into the AI-powered developer platform
GitHub is announcing general availability of GitHub Copilot Chat and previews of the new GitHub Copilot Enterprise offering, new AI-powered security features, and the GitHub Copilot Partner Program.

Written by	**Est. reading time**
Thomas Dohmke	9 minutes

Nov 8th, 2023

9 😄 1 👍 2 🎉 1

14 replies Last reply 5 months ago

Fig. 1. Reactions to a post in a NAV Slack channel

Zoom or Teams - supported the quality aspect by allowing for detailed discussions without overwhelming the main communication channels. Our observations of a large NAV Slack channel showed that it was frequently used for things such as sharing new research about relevant technology, tips on how to solve common issues, and discussing the use of new features. We saw that the use of specific emoticons as reactions to someone's posts, oftentimes uploaded GIFs not provided by Slack's regular emoticons, also added to the quality aspect. It not only provided a way to be creative and foster shared, organization-wide cultural expressions, but also a way for NAV employees to tailor precise reactions to their peer's utterings, avoiding misunderstandings. See an example in Fig. 1.

Another crucial point for ensuring quality communication across a large-scale organization consisting of many teams and dependencies, is the possibility of communicating quickly and to-the-point. The real-time nature of communication on Slack facilitated a degree of promptness that is difficult to get using alternative platforms such as email, and the use of Slack for frequent communication and support to other teams was a recurring theme in the interviews. One interviewee, P6, also told us how his team was using Slack for an easy and accessible way of note-taking during meetings: *"We have a good routine for sharing things that happen on Slack; we are good at writing minutes [in a team channel]. We write simple, short documents where we try to condense what we have actually agreed upon to develop, what the goal is, and how we want it in the team. We try to be more conscious about this documentation."* This shows how Slack's different options for use could inspire NAV teams to think innovatively with regards to how they implemented agile thinking into their everyday development process.

The rapid exchange of information also meant that the interviewees spent a lot of time on the platform, however. When asked about the time spent on Slack daily, P1 humorously responded, *"I wouldn't even want to know. But I guess in the course of an hour, maybe 15 min is on Slack."* Despite the benefits of Slack for communication quality and quick responses, then, the platform's user friendliness also came with an expectancy of developers in the organization "always" being available. P1 talked about how a life without Slack would have been "lovely" because he then would have more time to program. But he also reflected on how much his time spent on Slack helps others: *"I would probably have been more productive on paper. Because then I would have programmed a lot more. But I believe a consequence would be that someone else would have been less productive."*

P4 further elaborated on the frequent use of Slack, *"Our team has a dedicated Slack channel. I spend a lot of time on Slack even after work hours. It's essential for staying updated, but it can be overwhelming."* This quote was echoed by P9 who had "muted" several channels to not be disturbed when programming: *"There's always a lot of stuff going on [in the channels], and like... It's often just a distraction, I think, a lot of what's going on. It would probably be good to be shielded from it now and then. I think I've gotten better to just ignore it a bit."* This quote also points to that what may be useful to one NAV developer in one part of the organization, might be merely a disturbance to another employee in a different team. P13 had turned off notifications from other people but still received notifications from integrated bots: *"I only have notifications from Github, Dependabot, and pull requests. I get notified when someone opens a pull request where one of the dependencies is out of date. I wish there were fewer notifications, but I really appreciate knowing this immediately."*

4.3 Communication Discipline

We found that Slack's structure with channels and threads fostered a disciplined approach to communication and coordination that also facilitated ad-hoc communication. For instance, P5 said: *"We create threads for what we need, in addition to using huddle a lot for video. So, it's a common practice that one person starts a huddle, writes a topic, or adds a topic, and then whoever wants can join. It's not a meeting, it's not like you call people in. It can be a two-second agreement"*.

However, as touched upon in Sect. 4.1, although NAV's developers encouraged each other to conduct their conversations in open channels, not everyone was comfortable with this. As a consequence, some experienced developers took on the task of being available for questions and responding quickly: *"I often provide support in open channels [especially to] new employees and summer interns"*, remarked P1. He was among those of our informants who saw themselves as 'Slack mentors' for new hires and younger developers. Thus, for new employees, informal Slack conversations with more experienced developers were an important part of their onboarding process into NAV, seeing as Slack use was a given among the IT employees. P12, who had over 10 years of experience as a developer, explained that he had the impression that the younger employees, or "the chat generation" as he put it, seemed to appreciate that communication amongst the team primarily took place over Slack. He said: *"[Team communication] happens mainly on Slack, even if we are all present at the office. The exception is if we program together"*. This also meant that it was easier to go back and recall earlier conversations with colleagues, and keeping track of tasks.

That Slack use quickly became part of their everyday lives as NAV IT employees was also underscored by the younger interviewees. P2, who had only worked in the organization for one month, told us that although he had not received any informal introduction to using Slack upon starting working at NAV, he was 'learning by doing' and had already figured out many norms in regard to Slack use in the organization. When asked about the response time for answering people's messages he said: *"I reply within an hour. Or less. An hour is kind of a long time. It depends a bit on who it is, though."*

Although we were informed that guidelines for Slack use had been developed at some point and for certain channels, none of the developers we interviewed mentioned these. Nevertheless, it was evident that practices for use were ingrained in the organizational culture, and an integral part of learning the agile way of working in the organization. P5 contrasted Slack with other communication tools, noting a preference for Slack due to its ease of use and less formal nature, and hence making it easier to work in a disciplined manner: *"It's easier to join and leave conversations on Slack, which promotes open discussion."* That Slack use facilitated for disciplined working was underlined by P3, although he also included a word of caution with regard to choosing channels wisely: *"There are lots of different channels and you need to select with care. It can be overwhelming, so initially you might start the wrong place, but then you just get pointed in the right direction."* This strategy reflected a broader need among software engineers in the organization to balance accessibility and focused work.

5 Discussion

In this study, we investigated how software developers use the ESN tool Slack in a large-scale agile organization in the public sector. Previous studies have pointed to how several communication practices are crucial to alignment in large-scale agile software development, including ad-hoc, oral, formal, and informal communication [15,16]. The results of our study shed light on how Slack facilitated cross-organizational communication and coordination in NAV, but also in some ways created new challenges for the employees in the company. We now return to our research question: *How can the use of Slack facilitate or challenge organizational alignment in a public sector large-scale agile software development setting?*

While Slack offers a flexible platform for both formal and informal interactions, managing the balance between constant connectivity and productive work emerges as a key challenge. Our findings suggest that establishing clear expectations is needed in order to secure *communication transparency* across both the teams and the NAV organization as a whole, supporting the findings of Calefato et al. [11]. While it is crucial to expect employees to employ the same ESN tools in similar ways, such as being present in common channels, we found individual patterns of use such as a reluctance to post in open channels or respond to messages after work hours among some informants. Following Kostin and Strode, it can be argued that creating a safe environment for communicating transparently in agile teams is crucial [22], yet simultaneously some common user rules that may be tailored to individual teams would be beneficial for fostering a cross-organizational communication culture. As many teams were hybrid, meaning that team members were often working from home [33], this is especially a central point to consider.

In order to establish the best possible *communication quality* while using ESN tools, our findings point to how distinguishing between formal and informal information can be useful, carefully considering what information can be shared e.g., through an ad-hoc huddle, or would rather fit in writing in an open channel that can be found by others at a later point. By doing this simple exercise, cross-organizational communication can be enhanced, securing conciseness and clarity of the exchanged information. As some informants pointed out, important news and knowledge could get lost in Slack threads, and messages in some channels did not reach the right people. This corresponds to what Azarove et al. [4] underline; some people may not use Slack often, meaning that they might get distanced from the team's 'core' members as well as lose out on crucial information. It seems essential for employees, then, to establish clear guidelines on the appropriate usage of Slack, including posting important and formal information in allocated channels, at the right time, and using clear and concise language.

Lastly, to foster a common organizational *communication discipline*, employees should be given managerial support in ensuring optimal ESN use. As our study shows, using Slack had become part of everyday life as a developer in NAV. Although there existed some Slack user guidelines in the organization, these seemed to be not communicated to all developers, and it was thus up to each new employee to learn the Slack communication norms, perhaps with the help of self-appointed Slack mentors who again would have their own individual approach to Slack use. As the study by Stray et al. [36] underlines, there are clear differences in how experienced and less experienced developers utilize Slack, which could cause misunderstandings and lack of alignment in large-scale agile contexts. We recommend all new hires be taught and made aware of how to use ESN platforms both within the team and among the organization as a whole as part of the onboarding process. This highlights the critical role of managerial support in facilitating effective cross-organizational communication, which, in turn, would ensure a disciplined and consistent use of organizational ESN tools.

6 Conclusion and Further Work

In this study, we interviewed 13 developers working in a large-scale organization about their use of the ESN tool, Slack. We found that Slack was beneficial for communication transparency, as it fostered open and responsive communication among the developers. Slack facilitated transparent communication between employees, especially when the employees used open channels. However, some developers, especially new hires, avoided open channels in the beginning, which points to a need for onboarding developers in the communication norms and creating a safe environment. Overall, the interviewees were satisfied with the quality of communication on Slack. The platform's features, such as the use of threads, being able to edit messages after they are sent, and the ability to quickly share files or have conversations through a huddle supported the quality aspect by allowing for detailed discussions without overwhelming the main communication channels. The consistent and habitual use of Slack had also created clear patterns of communication discipline among the NAV's developers. However, the frequent exchange of information also meant that the interviewees spent a lot of time on the platform, both in fear of losing out on crucial information and due to the eagerness of participating in exciting conversations. It was thus clear from our findings that Slack was employed not only by necessity, but also due to fostering creative expressions and providing entertainment.

In future research, it is crucial to delve deeper into the team-level dynamics of utilizing ESN tools and examine how bots and integrations are implemented in such tools within large-scale agile settings. Additionally, finding the right balance between responding swiftly while avoiding constant distractions is a topic that requires deeper understanding, as many developers saw this as a central hurdle in their Slack use. Research suggests that developers are more adept at avoiding interruptions and managing them effectively when they do pair programming compared to solo programming [12]. Therefore, investigating how pair programming affects Slack use could provide valuable insights. Moreover, employees have a noticeable tendency to frequently check Slack, leading to potential distractions. Given that self-initiated task switching, such as checking Slack for updates, is found to be more disruptive than external interruptions, such as an impromptu huddle [1], this aspect warrants further investigation.

Acknowledgements. We would like to thank NAV for their engagement in our research. We are grateful to the interviewee participants for sharing their experiences and to Viggo Wivestad for helping with data collection. This work was supported by the Transformit project, partly funded by the Research Council of Norway under grant 321477.

References

1. Abad, Z.S.H., Karras, O., Schneider, K., Barker, K., Bauer, M.: Task interruption in software development projects: what makes some interruptions more disruptive than others? In: Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018, pp. 122-132. EASE 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3210459.3210471
2. Adolphe, L., Van de Zande, G.D., Wallace, D., Olechowski, A.: Analysis of virtual communication within engineering design teams and its impact on team effectiveness. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 83976, p. V008T08A038. American Society of Mechanical Engineers (2020)
3. Anders, A.: Team communication platforms and emergent social collaboration practices. Int. J. Bus. Commun. **53**(2), 224–261 (2016)
4. Azarova, M., Hazoglou, M., Aronoff-Spencer, E.: Just slack it: a study of multidisciplinary teamwork based on ethnography and data from online collaborative software. New Media Soc. **24**(6), 1435–1458 (2022)

5. Bablo, J., Marcinkowski, B., Przybylek, A.: Overcoming challenges of virtual scrum teams: lessons learned through an action research study. In: Stettina, C.J., Garbajosa, J., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming. XP 2023. LNBIP, vol. 475, pp. 34–49. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33976-9_3
6. Barbala, A., Sporsem, T., Stol, K.J.: A case study of continuous adoption in the Norwegian public sector. In: Proceedings of the 57th Hawaii International Conference on System Sciences (HICSS). Hawaii International Conference on System Sciences (HICSS) (2024)
7. Barbala, A., Sporsem, T., Stray, V.: Data-driven development in public sector: how agile product teams maneuver data privacy regulations. In: Stettina, C.J., Garbajosa, J., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming, vol. 475, pp. 165–180. LNBIP, Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-33976-9_11
8. Barbala, A.M.: Transcending Instagram: affective Swedish hashtags taking intimate feminist entanglements from viral to 'IRL'. Media Cult. Soc. **45**(1), 3–18 (2023). https://doi.org/10.1177/01634437221111930
9. Bernhardt, H.B.: Digital transformation in NAV IT 2016–2020: Key factors for the journey of change. In: Mikalef, P., Parmiggiani, E. (eds.) Digital Transformation in Norwegian Enterprises, pp. 115–134. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-05276-7_7
10. Broomfield, H., Reutter, L.M.: Towards a Data-Driven Public Administration: An Empirical Analysis of Nascent Phase Implementation, pp. 73–97 (2021)
11. Calefato, F., Giove, A., Lanubile, F., Losavio, M.: A case study on tool support for collaboration in agile development. In: Proceedings of the 15th International Conference on Global Software Engineering, pp. 11–21 (2020)
12. Chong, J., Siino, R.: Interruptions on software teams: a comparison of paired and solo programmers. In: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, pp. 29–38 (2006)
13. Conboy, K., Moe, N.B., Stray, V., Gundelsby, J.H.: The future of hybrid software development: challenging current assumptions. IEEE Softw. **40**(02), 26–33 (2023)
14. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. J. Syst. Softw. **119**, 87–108 (2016)
15. Dingsøyr, T., Bjørnson, F.O., Schrof, J., Sporsem, T.: A longitudinal explanatory case study of coordination in a very large development programme: the impact of transitioning from a first- to a second-generation large-scale agile development method. Empir. Softw. Eng. **28**(1), 1 (2022)
16. Edison, H., Wang, X., Conboy, K.: Comparing methods for large-scale agile software development: a systematic literature review. IEEE Trans. Softw. Eng. **48**(8), 2709–2731 (2021)
17. Ferreira, A., Antunes, P.: A technique for evaluating shared workspaces efficiency. In: Shen, W., Luo, J., Lin, Z., Barthès, J.-P.A., Hao, Q. (eds.) CSCWD 2006. LNCS, vol. 4402, pp. 82–91. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72863-4_9
18. Ghimire, D., Charters, S., Gibbs, S.: Scaling agile software development approach in government organization in New Zealand. In: Proceedings of the 3rd International Conference on Software Engineering and Information Management, pp. 100–104. ICSIM 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3378936.3378945

19. Henry, M.S.: The Use of Enterprise Social Networks for Social Support within Virtual Teams. Ph.D. thesis (2023)
20. Jackson, V., van der Hoek, A., Prikladnicki, R., Ebert, C.: Collaboration tools for developers. IEEE Softw. **39**(2), 7–15 (2022)
21. Janowski, T.: Digital government evolution: from transformation to contextualization. Gov. Inf. Q. **32**(3), 221–236 (2015)
22. Kostin, D., Strode, D., et al.: Effective communication in globally distributed scrum: a model and practical guidance. Australas. J. Inf. Syst. **27**, 1–42 (2023)
23. Lappi, T., Aaltonen, K.: Project governance in public sector agile software projects. Int. J. Manag. Proj. Bus. **10**(2), 263–294 (2017)
24. Lin, B., Zagalsky, A., Storey, M.A., Serebrenik, A.: Why developers are slacking off: understanding how software teams use slack. In: Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion, pp. 333–336. CSCW 2016 Companion, ACM, New York, NY, USA (2016)
25. Moe, N.B., Stray, V., Goplen, M.R.: Studying onboarding in distributed software teams: a case study and guidelines. In: Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering, pp. 150–159 (2020)
26. Moe, N.B., Stray, V., Hoda, R.: Trends and updated research agenda for autonomous agile teams: a summary of the second international workshop at xp2019. In: Hoda, R. (ed.) XP 2019. LNBIP, vol. 364, pp. 13–19. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_2
27. Mohagheghi, P., Lassenius, C.: Organizational implications of agile adoption: a case study from the public sector. In: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1444–1454. ESEC/FSE 2021, Association for Computing Machinery, New York, NY, USA (Aug 2021)
28. Nguyen-Duc, A., et al.: Work-from-home impacts on software project: a global study on software development practices and stakeholder perceptions. Softw. Pract. Exp. https://doi.org/10.1002/spe.3306. (in Press)
29. Parra, E., Alahmadi, M., Ellis, A., Haiduc, S.: A comparative study and analysis of developer communications on slack and gitter. Empir. Softw. Eng. **27**(2), 40 (2022)
30. Rahy, S., Bass, J.: Overcoming team boundaries in agile software development. J. Int. Technol. Inf. Manage. **29**(4), 1–31 (2021)
31. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**(2), 131–164 (2009)
32. Schulten, C., Nolte, A., Spikol, D., Chounta, I.A.: How do participants collaborate during an online hackathon? An empirical, quantitative study of communication traces. Front. Comput. Sci. **4**, 983164 (2022)
33. Smite, D., Christensen, E.L., Tell, P., Russo, D.: The future workplace: characterizing the spectrum of hybrid work arrangements for software teams. IEEE Softw. **40**(2), 34–41 (2023). https://doi.org/10.1109/MS.2022.3230289
34. Šmite, D., Moe, N.B., Šāblis, A., Wohlin, C.: Software teams and their knowledge networks in large-scale software development. Inf. Softw. Technol. **86**, 71–86 (2017)
35. de Souza Santos, R.E., Ralph, P.: Practices to improve teamwork in software development during the covid-19 pandemic: an ethnographic study. In: Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering, pp. 81-85. CHASE 2022, ACM, New York, NY, USA (2022)

36. Stray, V., Moe, N.B.: Understanding coordination in global software engineering: a mixed-methods study on the use of meetings and Slack. J. Syst. Softw. **170**, 110717 (2020). https://doi.org/10.1016/j.jss.2020.110717, https://www.sciencedirect.com/science/article/pii/S0164121220301564
37. Stray, V., Moe, N.B., Strode, D., Mæhlum, E.: Coordination value in agile software development: a multiple case study of coordination mechanisms managing dependencies. In: Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering, pp. 11–20. CHASE 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3528579.3529182
38. Stray, V., Moe, N.B., Vedal, H., Berntzen, M.: Using objectives and key results (OKRs) and slack: a case study of coordination in large-scale distributed agile. In: Proceedings of the 55th Hawaii International Conference on System Sciences (HICSS), p. 10 pages. Hawaii International Conference on System Sciences (HICSS) (2021). http://hdl.handle.net/10125/80225
39. Thompson, J.: A guide to abductive thematic analysis. Qual. Rep. **27**(5), 1410–1421 (2022). https://doi.org/10.46743/2160-3715/2022.5340, https://nsuworks.nova.edu/tqr/vol27/iss5/17
40. Walsham, G.: Interpretive case studies in IS research: nature and method. Eur. J. Inf. Syst. **4**(2), 74–81 (1995)

Coordination in Agile Product Areas: A Case Study from a Large FinTech Organization

Marthe Berntzen[1,2](✉), Silje Alette Engdal[1], Maja Gellein[1], and Nils Brede Moe[3]

[1] University of Oslo, Gaustadalléen 23B, 0373 Oslo, Norway
marthenb@ifi.uio.no
[2] Knowit Solutions AS, Universitetsgata 1, 0164 Oslo, Norway
[3] SINTEF, Strindveien 4, 7465 Trondheim, Norway
nils.b.moe@sintef.no

Abstract. Product teams organized into product areas are becoming more and more prevalent in large-scale agile. While such arrangements are thought to improve the development process and overall product delivery, it is still not clear how this form of organizing alleviates the coordination challenges commonly associated with large-scale agile. In this paper, we report on a case study from a product area in a large FinTech organization. Through analyzing interviews, observations, and strategic documents, we describe how organizing into a product area with vertical and horizontal teams supported inter-team coordination. Further, we describe seventeen coordination mechanisms used in the product area. Our findings have three main contributions. First, we propose that the product area represent a distinct organizational level that can support coordination in large scale development settings. Second, we found that the team types used in the product area represent different team typologies. Third, pull requests and pair programming were used as inter-team coordination mechanisms, and our findings suggest that pair programming to some extent could replace the PR mechanism to further improve product area coordination.

Keywords: Product areas · Product management · Large-scale agile · Coordination · Team typologies

1 Introduction

While the benefits of agile are well documented, large-scale agile product development settings are characterized by challenges with delivery speed, system complexity, and decision-making efficiency between interdependent development teams [3, 4, 11]. Such challenges are caused by inter-team dependencies that restrain development speed and progress [4, 19], which requires the use of coordination mechanisms. An inter-team coordination mechanism can be defined as an organizational process, entity, or arrangement used to manage dependencies to realize a collective performance [1].

Another way to reduce complexity and mitigate the coordination challenges caused by scale, is by establishing product areas where teams that work on similar products are

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 36–52, 2024.
https://doi.org/10.1007/978-3-031-61154-4_3

grouped together [16]. We understand a product area as a defined organizational structure that focuses on a particular product, or set of products related to a customer segment or business need [16]. It consists of product teams led by one or more product managers who ensure that the product meets both customer and business needs. Several product areas can operate independently within the same organization [23], which may allow for easier collaboration between interdependent teams, and reduce inefficient interfacing between unrelated teams [13, 16].

Studies conducted in recent years have focused on coordination within [32, 33, 36] and between agile teams [4, 10, 15, 22, 23] in software product development. However, despite that product management are commonly integrated with agile processes and large-scale agile frameworks [8, 23, 28], and that lack of alignment between teams and unclear dependencies are among the challenges with software product management [28], to our knowledge there is little research that specifically examine coordination in product areas. In this study we therefore investigate the following research question: "*How does coordination happen in product areas in large-scale agile?*".

2 Background

2.1 Organizing Large-Scale Agile Software Development into Product Areas

Agile development methods have been dominating software development, including large-scale settings, for the last decades [14, 24]. The benefits of agile methods, such as greater team autonomy, increased delivery speed and improved product quality have been well-documented in software engineering research [24]. Nevertheless, large-scale agile is characterized by coordination challenges and complex interdependencies that can reduce development speed and quality [4, 14].

'Large-scale' agile can be defined as two to nine teams, whereas 'very large-scale' agile involves more than ten development teams [12]. Such very large-scale cases require different coordination mechanisms, for example multiple inter-team stand-up meetings and retrospective meetings [10]. These mechanisms are needed to manage the many dependencies that arise from having many teams working together [1]. Establishing product areas where teams that work on similar products are grouped together [16], is one strategy to mitigate the coordination challenges in large-scale agile. The product area concept falls under the product management discipline, where 'the product' is placed at the center of the business goals and success [13].

Product management is an organizational discipline where the product is placed at the center of the business goals and success [17, 28]. In recent years, the term has been re-popularized by practitioners by writers such as Marty Cagan [7], Teresa Torres [35], and Christina Wodtke [37]. A recent systematic literature review on software product management [28] reports that, similar to large-scale agile [9, 14], the main challenges with software product management include problems with communication and synchronization across teams, problems with team autonomy and level of agility, and lack of clarity with regards to product management roles, especially the product manager [28]. Product managers perform activities related to product discovery and experimentation, strategic vision and product monitoring and adjustment, but also activities such as team support and stakeholder management [13, 34]. Today, many agile software development organizations use the product manager role in combination with the product owner role, which is typically understood as a narrower role [17, 34], but the terms are also used somewhat interchangeably [28]. Additionally, frameworks such as Large-scale Scrum (LeSS), has introduced specific roles such as the Area Product Owner [23].

Neither product management nor product areas are new terms within software engineering research [5, 13], but in recent years, the term has resurged also in the agile development research literature [21, 28, 34], perhaps due the focus of delivering customer value which is shared by both traditions [17]. However, as the inter-team coordination challenges that characterize large-scale development prevails, in this paper we explore coordination in product areas to understand how this type of organizational structure can support large-scale and very large-scale agile development needs.

2.2 Frameworks for Understanding Coordination in Large-Scale Agile

To understand coordination in product areas, it is necessary to understand coordination mechanisms used between teams. Recently, a taxonomy of inter-team coordination mechanisms in large-scale agile was developed, with the three main categories: *meetings*, *roles*, and *tools and artefacts*, and six sub-categories (see Table 1) [1]. These categories are commonly found in large-scale agile settings [e.g., 14, 22, 27, 31, 36].

Dependency management is core to coordination in large-scale agile. In line with the taxonomy of dependencies in agile software development, we define a *dependency* as something that occur when something, or someone, is dependent on the output of something or someone else [33], across three main categories: *Knowledge* dependencies are caused by the lack of information related to requirements, expertise, task allocation, or historical information. *Process* dependencies stems from the insufficient completion of development, or business process, activities, that blocks progress. Finally, *resource*

Table 1. The taxonomy of inter-team coordination mechanisms [1] (Color figure online).

Inter-team coordination mechanism: a process, entity, or arrangement contributing towards managing dependencies across teams	**Coordination meetings: Time-boxed arrangements with participants from different teams**	**Scheduled. Pre-defined meetings, with known participants and agenda, for example a stand-up meeting**
		Unscheduled. Ad hoc, based on the needs and availability of participants
	Coordination roles: Roles external to the defined development teams in the large-scale setting	**Individual**. Specialists and managers external to specific teams, such as architects
		Team. Specialist teams, such as task force teams
	Coordination tools and artefacts: Entities supporting development-related activities, or by-products of the development process	**Tangible**. Physical, touchable entities, such as a task board
		Intangible. Abstract, often digital, entities. For example, a product backlog

dependencies occurs when an object, be it a physical entity or a technical object, such as a piece of code, is needed for work to progress.

Dependencies require constant attention through coordination to avoid unnecessary delays and blockages that slows down progress [4, 19]. The dependency taxonomy has been adapted to the inter-team level [1–3] and used in several large-scale agile contexts [31, 32, 36]. To our awareness, however, neither of these theoretical approaches have been used at to study coordination in product areas.

3 Research Method

We explored our research question in a large FinTech organization [20]. We chose the case study research design because case studies can provide detailed insights to real-life settings where knowledge is limited [29], such as with product area coordination.

3.1 Case Description

The case consisted of 25 agile teams organized into several product areas. When we collected the data during early 2023, his way of organizing teams was relatively new for the organization, and they were still experimenting with the format. The product area we studied, the first of six product areas established in the organization, focused on personal banking, and had existed for about a year at the time. The product area consisted of nine teams with approximately 60 employees, where each team worked with features related to a specific product. In the FinTech organization a 'product' was defined as "a repeatable solution that can be offered to a market that solves a want or need" [Strategic document]. Moreover, the teams in the product area had shared overall goals, such as goals related to customer satisfaction and product area revenue. The product are teams were cross-functionally organized and consisted of developers, designers, team leaders, testers, and product managers, depending on the focus of the team. Team A-E (vertical teams in Fig. 1) worked with products such as establishing accounts and keeping overview of personal finances. Team F-G (horizontal teams in Fig. 1) worked with features that were shared by some or all the vertical products, such as bank accounts, payment solutions including credit cards, and the mobile bank app. The product area management team consisted of product managers, team leaders and representatives from development and technology, and design.

In terms of agile methods, the case organization did not subscribe to any specific scaling framework with pre-defined practices and coordination mechanisms. All teams could choose their own agile practices, which allowed for flexibility in practices like sprints, stand-ups, and retrospectives, which were used at varying frequencies by the different teams. In addition, the teams used the goal-setting framework Objectives and Key Results (OKR) in combination with practices from the Radical Focus framework such as the Monday Commitments and Friday Wins meetings [37]. Furthermore, all code needed to be checked by another developer to meet national standards for the financial sector. The formal software inspection process had been replaced by a mechanism called a pull request (PR), which was used within and across teams. In this approach, a contributor creates a PR after making code changes. Next, a reviewer inspects the suggested changes to see whether they can be merged into the base branch [18].

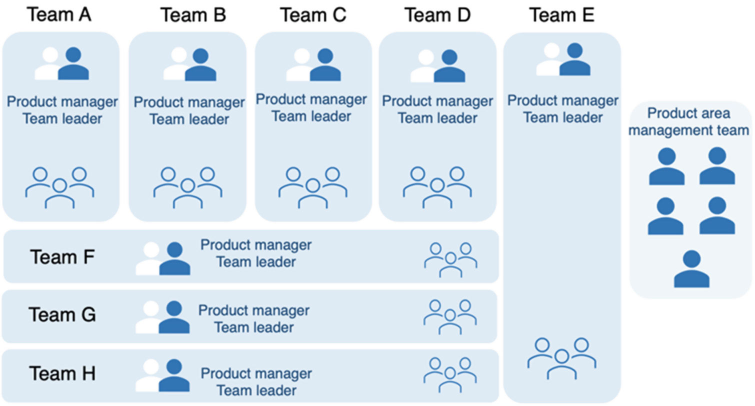


Fig. 1. The organization of the product area.

3.2 Data Collection

We collected our data from four of the product area teams (see Table 2 for an overview).

Interviews. Semi-structured interviews provided us with insights into how the product area was organized, which dependencies existed in the product area, and how they were managed. We conducted twelve interviews, where one of them was a digital interview with two participants. The other interviews took place in the organization's facilities. All interviews were tape-recorded based on participant consent, and then transcribed by the second and third authors. The average duration of the interviews was 50 min. We used an interview guide with questions about ways of working and inter-team collaboration such as *"Can you briefly explain the purpose of the product area?"* and "*On which parts of the product development do you collaborate across teams?"*.

Table 2. Data sources.

Data source	Details	Total
Interviews	Individual interviews with six developers, two designers, one tester, one product area manager, one tech lead, and a pair interview with two team leaders	12
Meeting observations	Five team status meetings, one product area meeting	6
Documents	Strategic documents and blogposts	6

Observations. We conducted field observations to gain a deeper understanding of the practices and provide additional insight to support the interviews. The observations were conducted by the second and third authors, who took notes during the meetings. We observed weekly meetings in five different teams in the product area, which allowed us to gain insight into internal team routines. Additionally, we observed an all-hands meeting (see Sect. 4.2) which provided insight into coordination across teams.

Documents. Strategic documents such as company presentations, role descriptions, and organizational blueprints, and organizational blogposts were used as supplemental data. These gave us a thorough understanding of the purpose with reorganizing the teams, as well as how the area was organized. The blogposts also contributed to enhance our comprehension of how they performed their work in the product area.

3.3 Data Analysis

The analysis was conducted by the first author, supported by the second and third authors who knew the data collection in detail, and the fourth author, an experienced qualitative researcher in the large-scale agile research field and who had been following the case for years [20]. We triangulated on the analysis by engaging in discussions about the findings to improve the credibility and dependability of our study [29].

We used coding procedures from thematic analysis [6]. Thematic analysis can be both used inductively and deductively [1, 6], and in this case our analysis was guided by existing the theoretical frameworks presented in Sect. 2.2. First, we used the taxonomy for inter-team coordination mechanisms [1] to categorize the mechanisms into meetings, roles, and tools and artefacts. Next, we analyzed which dependencies each mechanism managed using the dependency categories developed by Strode [33]. We also analyzed at which organizational level the mechanisms were used (Table 3).

4 Findings

4.1 Organizing Teams in Product Areas

Traditionally, the case organization had consisted of 25 development teams that were part of business areas that resembled a classical departmental structure. This very large-scale setting was characterized by many and varied focus areas and prioritizations, which made it difficult to deliver customer value with the desired speed and proved suboptimal in terms of aligning to resolve shared development needs: "*We realized that those who worked with payment cards, and those who worked with online payment had nothing in common. Although they both conceptually work with payment, they have no shared goals, their code is dissimilar … they work with completely different products.*" [I04, Product Area (PA) Manager].

As shown in Fig. 1, the teams were organized in vertical and horizontal structures. This way of organizing enabled the nine teams to better support each other and aligning the teams towards shared customer goals "*The way I see it, the product area is an attempt to gather teams that are related to each other. First and foremost, by having some level of shared goals, that are connected to our customers' everyday finances […]But it's also about how we want to be structured, with managers, product managers, team leaders, and how they collaborate*" [I02, Developer].

Working towards shared goals was described as a way of enabling dependency management: "*Customer value and business opportunities are often created where structures meet, and often there are dependencies between teams. Common goals across teams can be a way to remove blockages. It might be that one team need someone else to solve*

Table 3. Coordination mechanisms used in the product area, based on [1].

Coordination mechanism	Org. Level*	Description	Dependency managed [27]
Coordination meetings			
All hands meeting	P	A monthly meeting where teams share updates and align towards PA shared goals	Knowledge, process
Monday commitments	T	The team plans the week and decides on which goals to focus on	Knowledge, resource
Friday wins	T	The team reviews the week with a focus on celebrating achievements	Knowledge
Professional forums	P	Meetings among professional disciplines, often informal and held on a semi-regular basis	Knowledge, resource
Product management meetings	P	Manager-level meetings for discussions related to e.g., shared goals, strategic focus, and compliance in the product area	Knowledge, process, resource
Ad hoc meetings	T, P	Both unscheduled conversations and one-off meetings held to resolve tasks	Knowledge, resource
Coordination roles			
Product manager	T, P	Responsible for customer and business value in the product teams	Resource, process, knowledge
Team leader	T, P	Administrative and organizational focus, keeping teams up to date across the product area. Shield teams from "noise"	Knowledge, resource
Tech-lead	T, P	A developer with in-depth competence in the specific product. Has architectural overview, supports developers	Resource, knowledge

(*continued*)

Table 3. (*continued*)

Coordination mechanism	Org. Level*	Description	Dependency managed [27]
Product area management team	P	A product area level team with representatives from product management, development and technology, and design	Knowledge, process, resource
Task force team	P	Temporary team that works with specific features across teams	Resource
Coordination tools and artefacts			
Communication tools	T, P	E.g., Slack and Microsoft Teams. Enables communication and information sharing across teams regardless of location	Knowledge, resource
Documentation / visualization tools	T, P	E.g., Miro, Figma, JIRA and Confluence. Supports the development process and information sharing across teams	Knowledge, resource, process
OKR	T, P	Goal-setting framework used to set team-level goals and over all goals shared at the product area level	Knowledge, process, resource
Digital boards	T, P	Various digital displays of team-level or product area information	Knowledge
Pair programming	T, P	Developers working together in pairs at the same workstation	Resource, knowledge
Pull requests (PR)	T, P	A developer makes changes in another person or team's code, who subsequently asked review the change	Resource, process, knowledge
***Organizational level:** T = team level, P = Product area level			

something in order to progress. Then it is the product area's responsibility […] to make way". [I04, PA Manager]. Defining goals at the product area level was.

key: *"We have been able to bring them together, to get their goals aligned. We had not been able to do that with 25 teams"* [I04, PA Manager].

A key feature with the vertical and horizontal team set-up was the way roles were shared across teams. Although each team had a cross-functional basis set-up with developers, product manager and team leader, not all teams had all roles. This meant that sometimes teams needed to "borrow" expertise from each other within the product area: "S*ometimes the competence for testing the solutions is in a different team. So, then we need to coordinate a bit between teams if I have something that needs testing and then those who know how to test it will come and do it*" [I10, Developer]. Another explained that "*the designers have started to work a bit outside the team, they try to see across where they can be of help*" [I06, Developer]. Additionally, there were not a one-to-one correspondence between the team leaders, product managers and the development teams, which was explained as a benefit. "*We have created a flexibility in the area [...] We think more in terms of capacity than in the number of heads per team. It means that it is easier to say that 'okay, we need an additional team, you go in there as a product manager'*" [I04, PA Manager].

All in all, organizing the teams that worked with personal banking in a distinct product area enabled the teams to coordinate more efficiently than when they were part of the larger 25-team structure. In the following, we describe sixteen coordination mechanisms used in the product area. Table 3 contains the full list of coordination mechanisms, and Sects. 4.2–4.4, illustrate how they were used.

4.2 Product Area Coordination Meetings

We identified six coordination meetings. Two of these were conducted at the team level, and three at the product area level. One meeting occurred at both levels.

First, the monthly **all-hands meeting** contributed to shared overview across teams. "*We needed a shared understanding of the overarching goals we are working towards. At the same time, we need to tell each other about what's happening in the teams. So the all-hands meeting is actually a combo of a show-and-tell and our goals at a higher level*" [I04, PA manager]. Knowing who was working on what contributed to managing knowledge (task allocation) dependencies, and the focus on aligning towards shared goals contributed to managing process (activity) dependencies.

Additionally, various **professional forums** were held among the specific professional groups. For example, Android developers met every Friday. "*It's like a professional exchange, and a bit like if anyone have any problems they are working with, this is an arena where we can discuss it*" [I10, Developer]. As such, these forums contributed to managing knowledge dependencies related to the expertise of others, but also in identifying technical and resource dependencies that blocked team progress.

All teams held **Monday commitments** "*which is a check-in where everyone get together and share our goals for what we will achieve this week*" [I02, Developer] and **Friday wins** "*which is more like 'what did we actually accomplish' with cake or snacks and good vibes*" [I05, Developer]. Although these were team-level meetings, they worked as coordination mechanisms at the product area through the team leaders sharing information in the product management sync meetings. Moreover, "*all the Monday commitment boards are displayed in a shared Miro-board, so that everyone can see what we are doing all the way*" [I04, PA manager].

Finally, **ad hoc meetings** were widely used, and were held as needed. This enabled efficient dependency management: "*If something needs to be solved, it's basically just to invite people to a meeting to discuss it*" [I09, Developer].

4.3 Product Area Coordination Roles

We identified five roles, three individual and two team roles, specifically related to coordination across teams.

First, the **product managers** were domain experts responsible for the deliveries. They performed typical product owner tasks such as being the links between the development teams and the internal customers, and communicating requirements to the teams, but they were also engaged in more high-level strategic work: "*They are there to make sure we not only make something that is nice for the customer, but also creates value for the organization*" [I11, Tech lead]. They engaged in cross-team discussions about prioritizations and technical dependencies and "*often, it is the product managers that talk with each other, it is there the collaboration starts. And often it is them who puts together who needs to talk*" [I11, Tech lead].

Second, the **team leaders** were "*an administrative support function of sorts, who makes sure that we are onboard with what's happening in the other teams, in a coordinating role. Also, they shield us from external noise so that we can focus on what we're supposed to do.*" [I01, Developer]. As such, both product managers and the team leaders contributed to managing knowledge dependencies by acting as an information carrier across teams, as well as resource dependencies by making sure the teams had what they needed to proceed. Third, the **tech lead** role was involved with product area coordination by managing technical dependencies across teams. These roles were held by senior developers who "*have a bit more responsibility in the product area and attend discussions with tech leads from the other teams*" [I09, Developer].

The product managers, team leaders and tech leads, together with representatives from technology and design, were part of a **product area management team** who were responsible for overall themes related to shared goals, principles and vision, strategic and economic focus, product compliance, as well as team typologies in the product area. This team was "responsible for providing the teams with strategic context and make sure they have what they need to reach their goals" [Company presentation document], thereby enabling management of process and resource dependencies in the overall product development, and knowledge dependencies by supporting an understanding of shared goals. "*It is about facilitating […] to make sure that these teams have what they need to move quickly*" [I04, PA Manager].

Finally, if larger or more complex problems with technical and processual dependencies across teams needed to be resolved, the product area made use of **task force teams** which are temporary teams composed by members of different teams. These were used if "*there is a change that has a date, then we kind of have no choice. We have to put aside what else we were planning, and just do it. But that team will dissolve when the delivery has been made*" [I12, Team Leader].

4.4 Coordination Tools and Artefacts in the Product Area.

We found six tools and artefacts that contributed to inter-team coordination.

Several **communication tools** were used, including Slack, Microsoft teams for remote and hybrid meetings, e-mail for more formal communication. For example, Slack was used to communicate and coordinate both within and across the development teams through group channels and one-to-one messages, thereby managing dependencies across the product teams by shared overview and effective communication. "*A lot of it happens on Slack. We are often members of many team channels and product area channels, to see what is going on across the teams*" [I01, Designer].

Furthermore, usage of **documentation and visualization tools**, such as Miro, Figma, JIRA, and Confluence, contributed to managing dependencies by supporting cross-team collaboration and knowledge sharing within the team. Related to this, the **digital boards** were examples of artefacts produced by these tools that contributed to managing knowledge and process dependencies by providing easily accessible information about the other teams' goals or who was working on what: "*It is one large [Miro] board only, with canvases next to each other. It is very easy to just scroll sideways or downwards and zoom in to see what the other teams are doing*" [I01, Designer].

To express and measure progress towards the shared goals discussed in Sect. 4.1, the product area used **OKR,** a goal-setting framework that focuses on reachable business goals by formulating objectives for what to accomplish with corresponding key results that can measure what is actually achieved [37]. OKRs convey information about product and organizational goals, thus contributing to managing knowledge, resource, and process dependencies related to the overall product development. All teams formulated their specific OKRs which they revisited during weekly meetings and displayed in the digital boards. Additionally, "*we have OKRs at the organizational level, about four or five focus areas. From those, we have selected some to focus on, while other product areas focus on others*" [I09, Developer].

Another interesting coordination tool was **PR** used across teams. This contributed to managing technical resource, but also process and knowledge, dependencies in that team members could make changes themselves rather than waiting for others to become available. In addition to making a PR it was seen as important to engage in dialogue: "*Pull requests are really about creating things together. It's about talking together. And about saying 'Hi, I have some thoughts about your product from our perspective. What do you think about this, is this something we could do?'*" [I04, PA manager].

An alternative to PRs were to engage in **pair programming,** where developers work together. While pair programming was widely used within the teams, one explained "*if I work with a larger task, I will often engage in pair programming. Mostly with someone from outside the team*" [I09, Developer]. Another developer explained, "*It is easier to ask someone that works in the same area, the level of engagement is a bit different than if you ask someone that works with their own [team's] problem*" [I05, Developer]. As such, pair programming contributed to managing knowledge and technical and entity resource dependencies related to the expertise of others, as well as process dependencies, in that it removed the need to wait on others to finish reviewing a PR.

5 Discussion

In this study we investigated the research question "*How does coordination happen in product areas in large-scale agile?*". Our findings offered three main insights.

5.1 Product Areas as a Distinct Organizational Level

First, a product area can be understood as a distinct organizational level (Fig. 2). Research has shown that coordination can become problematic in very large development settings, as the number of teams increase, so does both systems complexity and the number of dependencies between teams [3, 4, 10]. Therefore, very large-scale development settings requires different dependency management than smaller settings [11, 12]. Our findings illustrated that the very large-scale setup (25-teams) was not manageable in terms of shared goals and successful product delivery, and that dividing into several product areas that could function relatively independent of each other, with their own mechanisms for inter-team coordination, was a successful approach. While inter-team coordination challenges are well-described in the current literature [e.g., 2, 4, 10], and studies has described mechanisms offered by large-scale frameworks, such as Area Product Owners in LeSS [23], little research attention has been directed at specifically examining coordination in product areas.

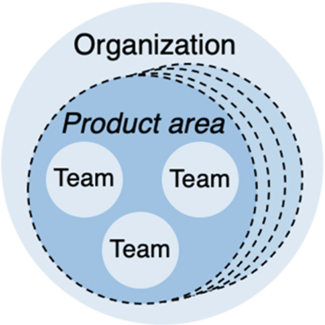


Fig. 2. The product area as a distinct organizational level.

We found seventeen mechanisms that contributed to managing dependencies in the product area. This number is comparable to other studies. For example, Dingsøyr et al. [10] describe a large-scale case of ten teams that used fourteen inter-team coordination mechanisms, Stray et al. describe nineteen mechanisms [32] and Vedal et al. [36] list 22 mechanisms from the same case with seven teams.

5.2 The Team Typology of Product Areas

Second, a key characteristic with the product area was the set-up of vertical and horizontal teams (see Fig. 1). This type of set-up is similar to what is described in the Team Typologies framework [25] which propose core team types including the stream-aligned team the platform team, the enabling team, and the complicated subsystem team. These types can be compared to the types of teams we observed.

First, *stream-aligned teams* are described as cross-functional teams who are independently responsible for building and delivering specific products [25]. The vertical teams are comparable to this team type as they focused on specific products, such as creation of bank accounts or products of financial overview. This comparison is in line with the general concept of software product teams [17, 28, 34].

Second, *platform teams* are typically internal teams that support and enable the steam-aligned teams in delivering the customer-facing products [25]. This compares to the horizontal teams that developed internal products shared by the vertical teams. Platform teams are often used in large-scale agile because they manage dependencies between the vertical teams by providing resources such as shared infrastructure [1, 10]. We also found that the horizontal teams provided designers and testers, as needed.

Third, *complicated subsystem teams* can be compared to the temporary task force teams described in Sect. 4.3, which were assembled to resolve specific priorities with high levels of complexity that required a greater extent of inter-team coordination. According to Skelton and Pais [25], most organizations will not need this type of team on a permanent basis. However, in line with our findings, other studies support that temporary task force teams are useful in large-scale software development when specifically complex task needs to be solved [1, 3, 22].

Finally, we compare the product area management team to the *enabling team* type. This team type assist, coach and support teams as needed [25]. In the product area, the management team helped setting and managing the product teams' goals and directions, removed barriers, and made sure they were provided with training in for example, product development practices and use of the OKR framework.

5.3 Pull Requests and Pair Programming as Coordination Mechanisms

Third, our findings shed light on how pull requests and pair programming were used as inter-team coordination mechanisms. While both these tools are well-known and well-described within software engineering, our study is among the first to show how they could be specifically used to manage dependencies between teams.

Use of the PR practice enable knowledge sharing and aim to balance the skills in the teams [18]. In our findings, the PR mechanism enabled solving dependencies within and between teams because instead of waiting for others to add, change or delete code, a developer could work in others' code and then create a PR. Pull request have been used as a tool to manage technical dependencies across teams in other large-scale settings too. For example, Šmite et al. found that the number of cross-team and even cross-tribe pull requests at Spotify is huge [26]. And while the PR mechanism is now standard for code reviews, challenges have been reported, such as when team members are distributed, or when teams in large-scale settings use different agile approaches [30].

Interestingly, we found that the use of pair programming to some extent could replace PR as a coordination mechanism, as developers from different teams writing code together eliminated the need to create, review, and approve PR across teams. However, we cannot say from these observations alone what constitutes an optimal balance between the mechanisms, or to what extent this finding can mitigate the challenges reported [30]. Future research should therefore investigate how PR are used as a coordination mechanism, and if, and when, pair programming could be used as a replacement.

5.4 Practical Implications

In addition to the theoretical contributions, our study offers several practical take-aways. First, we recommend managers of very large-scale product development settings to consider organizing into smaller product areas, preferably with fewer than ten teams, as this is considered a cut-off level of increased complexity [12]. Our findings show assembling a product area with teams that focused on the same goals, enabled the use of several shared coordination mechanisms, such as the Monday commitment and Friday wins meetings, and OKR at the product area level.

A second recommendation is to consider sharing resources within the product area. This provides greater flexibility in responding to changes, which is important to consider as coordination needs change over time [3]. Moreover, sharing roles and expertise within a product area enables more efficient management of resource dependencies, within the capacity limits that are set for the roles. We further recommend setting up a product area management team that can follow up on the longer-term strategic goals and support the product managers in facilitating goal attainment at the team level [34]. Finally, if the development context is characterized by widespread use of PR, we recommend using pair programming across teams as a replacement mechanism to resolve technical dependencies, as PR is a more passive action that requires inter-team communication which in worst case can create process dependencies and hold-ups.

5.5 Evaluation of Limitations and Research Quality

The main limitations of this study are the single-case study design, and the use of interviews as the main data source, which makes it difficult to generalize the findings [29]. However, we relied on several forms of triangulation using multiple perspectives to ensure rigor and quality in the research process and to clarify the findings [29].

First, we ensured data source triangulation by supplementing the interviews with observations and strategic documents. By interviewing participants from different teams, we gained access to participants' understandings of coordination in the product area. By observing meetings and supplementing with documents we obtained context and a deeper understanding of the participants statements during the interviews. Second, researcher triangulation was ensured by having different roles in the author team. The second and third authors collected the data, and the first author conducted the analyses. The fourth author contributed with additional data material and in-depth knowledge about the case organization. Third, methodological triangulation was used in the analytical phase, where we used the well-established thematic analytical framework [6], supplemented by the field-specific taxonomies [1, 33] which contributed to the relevance of the analysis in the large-scale agile empirical setting.

6 Future Research and Concluding Remarks

In this study we conducted a single-case study to understand coordination in large-scale product areas. This topic has received little research attention, at the same time as product management is re-surging as an important topic in agile software development practice. At the same time, the coordination challenges in large-scale agile prevails.

With this study, we have attempted to provide more knowledge on coordination in product management, by describing how coordination happened in a nine-team product area in a large-scale FinTech organization. We described seventeen coordination mechanisms used to manage dependencies, and illustrated how the product area can be represented as distinct organizational level. However, there is still a need for more knowledge on product area coordination, in particular in relation to what constitutes optimal product area size and team configuration [21]. Moreover, the product area organization form was relatively new for the case organization, and therefore, ongoing experimentation and changes were happening. With this study, we have only presented a snapshot, based on the data collection period. More research is needed to understand if and how changes in the product area also influence change in coordination mechanisms used in the product area. Additionally, we have described the team typology of product areas. Similar team types have been described in other studies, albeit not specifically within the product management context. This is an interesting avenue for future research. Finally, we identified pull requests and pair programming as related coordination mechanisms. However, we do not know enough about how these mechanisms can replace each other. Future research should aim to further advance our understanding of coordination in large-scale agile product management in general, and product areas in particular, with the aim of improving software product delivery and value.

Acknowledgments. The authors thank the informants for their willingness to share their experiences. This research was supported by the Research Council of Norway through the Transformit (grant no. 321477) and the 10XTeams (grant no. 309344) projects.

References

1. Berntzen, M., et al.: A taxonomy of inter-team coordination mechanisms in large-scale agile. IEEE Trans. Softw. Eng. **49**(2), 699–718 (2022). https://doi.org/10.1109/TSE.2022.3160873
2. Berntzen, M., Stray, V., Moe, N.B.: Coordination strategies: managing inter-team coordination challenges in large-scale agile. In: Gregory, P., Lassenius, C., Wang, X., Kruchten, P. (eds.) XP 2021. LNBIP, vol. 419, pp. 140–156. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78098-2_9
3. Berntzen, M., et al.: Responding to change over time: a longitudinal case study on changes in coordination mechanisms in large-scale agile. Empir. Softw. Eng. **28**, 114 (2023). https://doi.org/10.1007/s10664-023-10349-0
4. Bick, S., et al.: Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. IEEE Trans. Software Eng. **44**(10), 932–950 (2018). https://doi.org/10.1109/TSE.2017.2730870
5. Bosch, J.: Product-line architectures in industry: a case study. In: Proceedings of the 21st International Conference on Software Engineering, pp. 544–554 (1999). https://doi.org/10.1145/302405.302690
6. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qual. Res. Psychol. **3**(2), 77–101 (2006). https://doi.org/10.1191/1478088706qp063oa
7. Cagan, M.: Inspired: How to Create Tech Products Customers Love. John Wiley & Sons, Hoboken (2017)
8. Digital.ai: 16th Annual State of Agile Report (2022). https://digital.ai/resource-center/analyst-reports/state-of-agile-report/. Accessed 05 May 2024
9. Dikert, K., et al.: Challenges and success factors for large-scale agile transformations: a systematic literature review. J. Syst. Softw. **119**, 87–108 (2016). https://doi.org/10.1016/j.jss.2016.06.013

10. Dingsøyr, T., et al.: A longitudinal explanatory case study of coordination in a very large development programme: the impact of transitioning from a first- to a second-generation large-scale agile development method. Empir. Softw. Eng. **28**(1), 1 (2022). https://doi.org/10.1007/s10664-022-10230-6
11. Dingsøyr, T., et al.: Agile development at scale: the next frontier. IEEE Softw. **36**(2), 30–38 (2019). https://doi.org/10.1109/MS.2018.2884884
12. Dingsøyr, T., Fægri, T.E., Itkonen, J.: What is large in large-scale? a taxonomy of scale for agile software development. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (eds.) PROFES 2014. LNCS, vol. 8892, pp. 273–276. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13835-0_20
13. Ebert, C.: The impacts of software product management. J. Syst. Softw. **80**(6), 850–861 (2007). https://doi.org/10.1016/j.jss.2006.09.017
14. Edison, H., et al.: Comparing methods for large-scale agile software development: a systematic literature review. IEEE Trans. Softw. Eng. **48**(8), 2709–2731 (2022). https://doi.org/10.1109/TSE.2021.3069039
15. Gustavsson, T., et al.: Changes to team autonomy in large-scale software development: a multiple case study of Scaled Agile Framework (SAFe) implementations. Int. J. Inf. Syst. Proj. Manag. **10**(1), 29–46 (2022). https://doi.org/10.12821/ijispm100102
16. Heck, P., et al.: A software product certification model. Softw. Qual. J. **18**, 37–55 (2010). https://doi.org/10.1007/s11219-009-9080-0
17. Kittlaus, H.-B.: Software product management and agile software development: conflicts and solutions. In: Maedche, A., Botzenhardt, A., Neer, L. (eds.) Software for People, pp. 83–96. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31371-4_5
18. Maddila, C., et al.: Nudge: accelerating overdue pull requests toward completion. ACM Trans. Softw. Eng. Methodol. **32**(2), 1–30 (2023). https://doi.org/10.1145/3544791
19. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Comput. Surv. (CSUR). **26**(1), 87–119 (1994). https://doi.org/10.1145/174666.174668
20. Moe, N.B., et al.: Attractive workplaces: what are engineers looking for? IEEE Softw. (2023). https://doi.org/10.1109/MS.2023.3276929
21. Moe, N.B., et al.: Software product management in large-scale agile (2023)
22. Moe, N.B., et al.: To schedule or not to schedule? an investigation of meetings as an inter-team coordination mechanism in large-scale agile software development. Int. J. Inf. Syst. Proj. Manag. **6**(3), 45–59 (2018). https://aisel.aisnet.org/ijispm/vol6/iss3/4
23. Paasivaara, M., Lassenius, C.: Scaling scrum in a large globally distributed organization: a case study. In: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), pp. 74–83 IEEE (2016). https://doi.org/10.1109/ICGSE.2016.34
24. Palopak, Y., et al.: Knowledge diffusion trajectories of agile software development research: a main path analysis. Inf. Softw. Technol. **156**, 107131 (2023). https://doi.org/10.1016/j.infsof.2022.107131
25. Skelton, M., Pais, M.: Team topologies: organizing business and technology teams for fast flow. In: Revolution (2019)
26. Šmite, D., et al.: Decentralized decision-making and scaled autonomy at Spotify. J. Syst. Softw. **200**, 111649 (2023). https://doi.org/10.1016/j.jss.2023.111649
27. Šmite, D., et al.: Spotify guilds: how to succeed with knowledge sharing in large-scale agile organizations. IEEE Softw. **36**(2), 51–57 (2019). https://doi.org/10.1109/MS.2018.2886178
28. Springer, O., Miler, J.: A comprehensive overview of software product management challenges. Empir. Softw. Eng. **27**(5), 106 (2022). https://doi.org/10.1007/s10664-022-10134-5
29. Stake, R.E.: Qualitative case studies. In: Denzin, N., Lincoln, Y. (eds.) The Sage Handbook of Qualitative Research. Sage Publications, Thousands Oaks (2005)

30. Stray, V., et al.: An empirical investigation of pull requests in partially distributed BizDevOps teams. In: 2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE), pp. 110–119. IEEE (2021). https://doi.org/10.1109/ICSSP-ICGSE52873.2021.00021
31. Stray, V., et al.: Coordination value in agile software development: a multiple case study of coordination mechanisms managing dependencies. In: Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering, pp. 11–20. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3528579.3529182
32. Stray, V., et al.: Dependency management in large-scale agile: a case study of DevOps teams. In: Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS 2019), vol. 2019, pp. 7007–7016 (2019). http://hdl.handle.net/10125/60137
33. Strode, D.E.: A dependency taxonomy for agile software development projects. Inf. Syst. Front. **18**(1), 23–46 (2016). https://doi.org/10.1007/s10796-015-9574-1
34. Tkalich, A., Ulfsnes, R., Moe, N.B.: Toward an agile product management: what do product managers do in agile companies? In: Stray, V., Stol, K.-J., Paasivaara, M., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming: 23rd International Conference on Agile Software Development, XP 2022, Copenhagen, Denmark, June 13–17, 2022, Proceedings, pp. 168–184. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-08169-9_11
35. Torres, T.: Continuous Discovery Habits: Discover Products that Create Customer Value and Business Value. Product Talk LLC (2021)
36. Vedal, H., Stray, V., Berntzen, M., Moe, N.B.: Managing dependencies in large-scale agile. In: Gregory, P., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming – Workshops: XP 2021 Workshops, Virtual Event, June 14–18, 2021, Revised Selected Papers, pp. 52–61. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-88583-0_6
37. Wodtke, C.: Radical Focus: Achieving Your Most Important Goals with Objectives and Key Results. Cucina Media LLC (2021)

Software Product Management in Large-Scale Agile

Nils Brede Moe[1](✉), Marthe Berntzen[2,3], Astri Barbala[1], and Viktoria Stray[1,2]

[1] SINTEF, Strindveien 4, 7465 Trondheim, Norway
{nils.b.moe,astri.barbala,viktoria.stray}@sintef.no
[2] University of Oslo, Gaustadalléen 23B, 0373 Oslo, Norway
marthenb@ifi.uio.no
[3] Knowit Solutions AS, Universitetsgata 1, 0164 Oslo, Norway

Abstract. Large-scale agile software development is increasingly being organized with product management. Although product management is familiar to software engineering, we need research-based knowledge about organizing product management in an agile development context. This study focuses on product management challenges and configurations in a Nordic fintech organization with 10,000 employees. We conducted 19 interviews with participants involved in core product management activities. The study identifies ten key factors that hinder product management performance in a large-scale agile product company. Further, we present six product management configurations used in the fintech organization. The most suitable setup depends on the product lifecycle stage, product size, and development team setup. Our summary of the product management configurations and challenges can guide software product managers working in large-scale agile companies.

Keywords: Product Management · Team · Product Manager · Product Owner · Agile Software Development · Coordination

1 Introduction

Already at the large-scale workshop at XP2013 [9], how to manage complex and large development efforts with many teams was much debated. The topic is still very trendy as agile methods have become the de facto standard for product development in large companies. Some argue that a common large-scale agile framework like SAFe [18] and LeSS [17], is needed to define the roles, processes and artifacts. Others argue that context-based agile tailoring is vital to capture and address each organization's unique coordination context [6] and changes in coordination needs over time [3, 7], therefore introducing frameworks has a limited effect. Others again say that the most critical success factors are cross-functional teams that take responsibility for their work and coordinate, communicate, and align their actions with others. Spotify serves as an example of this [25]. A recent trend is to design teams to match the software architecture and give these teams a defined topology to reduce their cognitive load [24].

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 53–69, 2024.
https://doi.org/10.1007/978-3-031-61154-4_4

Regardless of the large-scale agile approach chosen, there needs to be a function that is responsible for defining and prioritizing what goes into the product, and that handles all the conflicting priorities [20] that exist in a large-scale context. If the teams do not have an up-to-date understanding of what to deliver and why, reducing cognitive load, increasing autonomy, or applying a framework does not help. In such cases, there may be a need for Software Product Management (SPM). SPM can be defined as the "organization and coordination of all activities important for a software product in order to achieve product success" [32]. In a recent study, Berntzen et al. [2] found the use of seventeen coordination mechanisms in a product area with 8 teams. Therefore, we argue that SPM plays a critical role in large-scale agile. SPM involves understanding market needs, defining product vision, and working closely with cross-functional teams to ensure successful development, launch, and ongoing management of the product [27]. SPM ensures a balance between technical and business perspectives and involves coordination between product stakeholders [11, 13]. The idea of SPM also fits well with the agile approach because a core SPM activity is to ensure the interest of the customers and clients of the product [27].

Product management means being responsible for a defined product [11]. In large organizations, such as a bank, a product can be a customer onboarding experience, a credit line, advisory services, or a banking account. When handling a small product, the product owner can handle SPM. When the product grows, a team of product owners can take on this job [23], often in combination with one or several product managers. Different products require different expertise and team configurations and, therefore, a different organizational setup, i.e., different product management configurations. However, research on how product managers' roles and practices are configured in large-scale agile settings is limited. There is a distinct lack of comprehensive understanding of the specific organizational and team structures that support effective product management, particularly in organizations with many thousand employees that deliver both B2B and B2C products.

This study seeks to address this knowledge gap by investigating the challenges and configurations of product management within a large-scale agile organization. Specifically, we aim to answer the following research questions:

- RQ1*: Which challenges do product management roles face in large-scale agile?*
- RQ2: *What types of product management configurations exist in large-scale agile?*

To investigate these questions, we conducted a case study of DNB - a Nordic agile fintech organization with 10,000 employees where 130 people hold product management roles such as product managers and product owners. Through this inquiry, we aim to shed light on the organizational and team setups that facilitate effective product management, contributing to a more nuanced understanding of agile practices at scale.

2 Background

2.1 Large-Scale Agile and Product Management

In very large-scale agile companies, the complexity of the software development effort is enormous as many stakeholders are involved, often with conflicting interests. As a result, the need for inter-team coordination increases significantly [14]. Further, the products must adapt to constantly changing user needs, which requires an end-to-end flow between customer demand and the fast delivery of a product or service [12], which is challenging. Mikalsen et al. [20] found that agile product teams in a bank needed to negotiate with other teams, business units, and key management stakeholders to ensure that the digital offerings made sense in terms of both users' needs and company revenue. Therefore, SPM needs to scale outside of the software development department and link to other functions such as marketing, sales, and operations.

Despite the importance of SPM, few empirical studies are focusing on SPM in large-scale agile [15, 16]. The current literature, including studies by Ebert [10], Tkalich et al. [31], and Springer et al. [27], have explored various facets of software product management, from role definitions to acquiring development resources and technical debt. Paasivaara et al. [23], highlights several challenges related to product management configurations and roles in a company during an agile transformation, such as challenges with defining the product owner (PO) role. Scaling the PO role is a common SPM practice. However, the PO role is often not enough, and additional SPM roles and configurations are often needed. Sometimes there is a team of POs [4], or product managers (PM) that lead the products, and that balance software development, people and politics [11]. Ebert and Brinkkemper [11] highlight the risks of operating without a dedicated PM, pointing to issues such as diluted leadership, suboptimal performance, increased rework, and delays. This underscores the critical need for focused product management to overcome the complexities of large-scale software development.

SPM has been found to have different setups in large-scale agile. It is common to organize into product areas where teams work with a common product or sub-product. A typical set-up within a product area can be one or more development teams, one or several POs and a PM that all work with the same product(s). Berntzen et al. [4] studied the development of a public transportation platform with 13 teams and 9 POs. Seven of the POs had one team, whereas two had three teams each. The weekly PO coordination meeting and quarterly workshops were facilitated by a PM. A much more complex setup was found by Smite et al. [26]. They describe a Telecom product at Ericsson developed by seventeen self-managing cross-functional feature teams; five teams in Sweden, ten teams in China, as well as two Korean teams. The teams were supported by technical experts, line managers, product owners, operative product owners, system managers, configuration managers, testing framework experts, continuous integration experts, agile coaches, and integration leaders. SPM becomes complex in such a setup. The authors conclude that networking is the primary mechanism when solving complex tasks in large-scale product development.

2.2 Challenges with Software Product Management

Introducing and succeeding with SPM takes time because of the many challenges that are associated with software product development. The larger the organizations and the longer the product life cycles, the more challenging. Ebert and Brinkkemper [11] studied a business unit producing components used in communication networks. The unit operated in North America, Europe and Asia, and developed platform products that are customized for contract projects. They found that results from introducing SPM were first available for the business unit after 12–18 months of working with a new

SPM scheme. Therefore, to succeed with SPM it is necessary to be aware of the main problems that affect SPM and to provide solutions for dealing with them. Springer et al. [27] organized 15 focus groups with 47 software product managers to understand their challenges. The 5 most common problems were:

- Determining the true value of the product that the customer needs.
- Product managers must work iteratively with teams to understand the customer needs and scaling opportunities of the product.
- Strategy and priorities are changing frequently, which makes product managers and their teams struggle with prioritization.
- Technical debt slows down the product development process and makes it difficult to prioritize the product roadmap.
- Working in silos. Initiatives that run across different departments require the need to align teams around common goals and synchronize them.
- Balancing reactive and proactive work. Mature products struggle to prioritize innovation and new customer value against bug fixing and maintenance work.

Agile methods might mitigate many of these problems. However, Springer et al. [28] found that a core challenge is that teams are not agile, they just follow rules and do not use experimentation and learning, which is essential for succeeding with SPM.

3 Research Method

We chose a case study approach [29] to study SPM in DNB, which is a fintech organization in the Nordics. A case study provides in-depth and detailed knowledge, which is fitting for this study as there is little research-based knowledge about product management in large-scale agile. For five years, the first and fourth authors collaborated with one of the organization's business units on team autonomy, product development onboarding, and distributed work [21]. The results of this previous collaboration were why DNB wanted a research-based approach to improving their SPM. In particular, there was a need to understand challenges, various configurations, and responsibilities. We interviewed people involved in product management from five business areas to understand the challenges of product management in large-scale agile and to understand which product management configurations exist. The three first authors conducted the interviews, authors two and three were responsible for the analysis, and all authors participated in meetings with the company.

3.1 Case Description

DNB has 10,000 employees mainly distributed over three cities. Out of these, over 130 people are involved in core product management activities and most hold a product manager or product owner role. DNB can thus be classified as very large-scale agile [8]. DNB did not subscribe to any specific scaling framework with pre-defined roles. All teams could choose their own agile practices, which allowed for flexibility in practices like sprints, stand-ups, and retrospectives. The organizational environment is characterized by both structural and technological changes. Over 70% of the POs and PMs had more than 5 years of experience in the company. They all had varied backgrounds; some had a technical (e.g., engineering) background and had been working in the product domain for several years, while others came from the business side. A few also came from other industries. The company introduced the PO role in 2017.

3.2 Data Collection

We interviewed 19 people involved in product management activities (Table 1). Each interview was held on Microsoft Teams and was recorded and automatically transcribed based on participants' consent. Most interviews were conducted by two of the researchers, while two were conducted by the first author only. The interviews lasted from 46 to 70 min (55 min on average). We used an interview guide to steer the conversations and make sure we touched upon topics of interest but allowed the conversations to develop naturally. Core questions were: "Describe the technical and business aspects of the product", "What works well/not well in your job?", "How are decisions made?" and "What competence is important in your role?".

In addition, we organized meetings with additional managers to understand the history of product management in the company and the context that the interviews were part of, e.g. department structure, the relationship between departments, products handled by each business unit, important company milestones, technical platform, and so on. We wrote minutes from these meetings. We also presented the results back to the organization in several meetings to verify our findings and to clarify misunderstandings.

Table 1. Data sources.

Data source	Details	#
Interviews	Product managers (5 female, 4 male)	19
	Product owners (5 male)	
	Other product development role (2 female, 3 male)	
Meetings	Managers and people responsible for improving product management	5
Documents	Strategic documents, product management survey, annual reports	3

3.3 Data Analysis

We used a thematic analytical framework [5] to analyze the data, which has been extensively described for software engineering research [1]. Thematic analysis proceeds through six iterative phases, where it is possible to move back and forth as the analysis develops [1, 5]. First, we divided the interviews among the authors, and read through each of them using open coding techniques to familiarise ourselves with the data and generate initial codes. From this, themes started to form, such as tasks and role descriptions, and challenges with product management at DNB. Next, we reviewed the theme by discussing each interview together to reach an agreement on categories and themes in the data. As part of reviewing the themes, we performed a second analysis, digging deeper into the data to better understand the challenges product managers and product owners experienced at DNB. The challenges identified from each interview were coded first at a lower-level category before similar codes were grouped together as themes. During this phase, we discovered that there were different product management configurations across the product areas. We therefore performed a second round of coding where we focused on understanding the different configurations present in the data. In the last phases of the analysis, we focused on refining themes by defining naming and mapping the different configurations to reach a coherent set of findings presented in the next section.

4 Findings

4.1 Software Product Management in DNB

In DNB, a 'product' was defined as something that delivers value, at the intersection of customer needs, business objectives and technology enablement. This meant that most things that deliver value to the customers could be defined as a product, e.g., a customer onboarding experience, a credit line, advisory services, or a banking account. Product management meant working closely with the cross-functional agile team, being accountable for the long-term business value of the product, and management of the product (including strategy, design, technology, risk, pricing, compliance, etc.) through the whole product lifecycle.

We found three key product management roles. First, the PM was described as "the CEO of the product". This meant being responsible for vision and strategy, roadmap and high-level priorities, and business outcomes throughout the product lifecycle. Second, the PO focused on hands-on work with engineers and designers, transformed the high-level vision of the PM into epics and an updated backlog and enabled agile ways of working. The third role, often called 'product lead' (the role was not yet formally named) was used when there was a need to facilitate portfolio alignment across product teams or alignment between product areas where one PM was not enough.

When investigating how to succeed with product management across these roles, three characteristics stood out: 1) Having a large network and knowing who knows what, 2) working closely with the team, and 3) working as an internal diplomat in the organization. This included focusing on setting aside time to listen, discuss, and find solutions through informal meetings. One explained the importance of knowing the company and stakeholders in order to succeed in the role: *"My biggest, what do you call it, badge is the fact that I know how the organization is built. I know who to go to, and who not to go to. And DNB is complex in that regard"* [I15]. Another continued: "*There are lots of experts and good people everywhere. You just have to know where to ask"* [I08]. One interesting observation was that teams seemed to change often, and many did not know exactly how big their product team was. One PM said: *"I haven't counted, but I think we're around maybe currently seven or eight*" [I13].

Next, we present the main challenges with product management. Then we report on the software product management configurations that we found in the interviews.

4.2 Challenges with SPM in Large-Scale Agile Organizations

We found 10 challenges reported by more than five of the 19 respondents (Table 2). Some of the challenges were related to the very large-scale nature of the organization. A PM explained challenges caused by the size (10,000 employees) (C4): *"The way we are set up is a bit… It is not easy to get the end-to-end value chain responsibility carried out. Because there are so many areas that play their part in the value chain. And perhaps they must do that. Because there are such complex systems and such complex value chains involved"* [I06]. A department manager explained that the prioritization challenges (C2) were also related to the company size: "*You get different steering signals, which means you don't necessarily get the speed you want. Or that there are an awful lot of people you must talk to before you can get anything done"* [I08].

Further, there appeared to be a connection between some challenges and some types of SPM configurations. For example, a PM who did not have a PO in the team reported having too diverse work tasks, taking on PO tasks, and spending too much time in meetings (C1): "*One moment you are working with risk management, with deadlines and set timeframes. In the next, you are working with the team as part of a creative and problem-solving task force. Then, you need to follow up on things that should be discussed [with stakeholders] and get a decision. It is a very wide task range*" [I02]. In a related manner, a PO who did not have a PM on their team and who was dependent on another part of the organization for technical resources (i.e., developers) explained that the shortage of resources limited their freedom to act (C6): "*Capacity is always a thing. It can be difficult to coordinate when you lack the resources*" [I05].

Another interesting challenge (C3) was related to insufficient user data management, that is the collection and analysis of data generated from DNB internal and external users. Codes within this challenge included a lack of focus on working with data-driven insights in SPM: "*That demand has never been on our plate. I would say nobody has asked us to, you know, take on this task*" [I17], and not having the resources or opportunity to fully utilize the value of data-driven insights: "*I have access to a system, but I have never had the time to fully understand how I can use it to get the data I actually need to make a proper analysis*" [I06].

As described above, a key point to succeeding with SPM in DNB was the ability to network and get others to follow. Moreover, the many prioritizations to be managed (C2) and at times challenging organizational structure (C4) as well as the need to involve top management in many decisions (C7) made stakeholder management (C9) a challenge in DNB. We found that those who struggled with stakeholder management often lacked a large internal network or struggled to understand the inner workings of the organization: "*I have underestimated the importance of understanding how big our organization is, and how important the stakeholder part is. You can come from a smaller company, […] and know the [product management] theories very well. But that does not mean that it works in DNB"* [I14].

Table 2. Challenges with product management in the case organization

Challenge (C)		Description	#
1	Too many tasks and responsibilities	Too many diverse tasks with urgent priority, too much time spent in meetings, too little staffing to fill all tasks, and having to do the tasks of other SPM roles, unclear division of responsibilities between roles	17
2	Prioritization challenges	Prioritizations come from many different parts of the organization, and many are unclear	15
3	Poor data management	Gaining value from data, such as not enough data being collected, having poor data management tools, and a lack of personnel working with data and data insights	11
4	The company structure	The way the organization is currently set up does not provide ideal conditions to work with SPM	9
5	Lack of customer focus	Goals from business rather than customer value, not sufficiently collecting or using customer feedback, then difficult to placing customer wants and needs first	7
6	Lack of freedom to act	Challenges with freedom to act due to dependencies between products and shortage of resources, both human and technical resources and dependencies	7
7	Upper management is not involved enough	Upper management makes prioritisations and decisions on SPM but lacks knowledge and insight about SPM	6
8	Unclear goals and reporting of goals	A lack of a unified way of working with goals across the company, such as the use of KPIs, OKRs, or others	5
9	Stakeholder management	Knowing whom to involve, when, having an efficient internal network, and spending too much time on stakeholder management	5
10	Lack of alignment between organizational units	Somewhat related to challenge no. 4, the different ways of working between the DNB business units caused challenges with aligning SPM-related activities	5

# = The number of participants referring to the challenge

4.3 Product Management Configurations

We found a range of SPM configurations, and the set-up of product roles and teams depended on the nature and lifecycle of the product. As there were conflicting priorities

(C1, C2, C10), not all products had an ideal team set-up or SPM role configuration to match their specific product needs. Figure 1 shows the configurations described.

Configuration A: The PM and the PO is the Same Person, Part of One Team
This configuration was found in several products with different characteristics, often when the product was small in terms of the number of developers and stakeholders involved, or when the product was in the early phases (before scaling). One PM who worked on a new product relating to customer programs and covered both the PM and PO roles for her team, at least for the time being, described: "*Now that it's so early [in the product phase], I have both roles. But I see that the more we develop… That is, in the initial phase, when you start things up, then it works. But the more you get into the operational side, there will be a need to get a product owner, too"* [I11].

In other cases, this configuration was described as not having a PO, and that the PM also performed the PO tasks. For example, in a large cross-functional team that worked with a product that allowed customers to gain an overview and insight into their personal finances, one PM said: "*I have long felt that I do not need [a product owner]. But it depends a lot on the team's structure, maturity, and tech stack. In some cases, it is necessary to be closer and coordinate on a lower level to get the details right.*" [I09]. However, also in this configuration, PO tasks had to be performed: "*Now that the product must be both maintained and further developed, I am feeling the need to have a product owner […] Perhaps I have been a product owner myself the past few years without reflecting on it*" [I09].

One PM who worked with a platform with both internal and external users explained that working on a complex product that involved a lot of strategic work was challenging and that therefore, there was a need for a PO to be more involved in the operational work. He explained: *"In my team, we only have me as a senior product manager. Optimally, I think we would have two, one senior and one that is more junior, which then we could split – some teams do it, though, split responsibilities, right? So, one is more operational, the other one is more strategic".*

Configuration B: One PM and One PO are Part of One Product Team
In this configuration, a PM and a PO were together responsible for one product team. For example, in a product team developing a system that enabled customers to invest money in companies. The PM of this team explained that he collaborated frequently with his manager, the PO, and the team (of which all six developers were located in another city) through daily meetings. The PM explained [I16]: "*We have digital tools and we have Teams and cameras, it is just like sitting next to each other*." Working with a small data-driven agile team was an enabler for development speed. He explained: "*Instead of spending a lot of time on planning and doing months of interactions [with customers] before we release a new product, we think that it is better to do the groundwork and get the product out and then rather make adjustments based on customer feedback".* However, the setup was not ideal: "*we have two backend developers and four frontends, so maybe the team is not balanced enough. Because we often see that things stop because you are waiting for the backend. It's something we've addressed above [to management] as well".* The PM's line manager played an important role. The PM explained: "*He had*

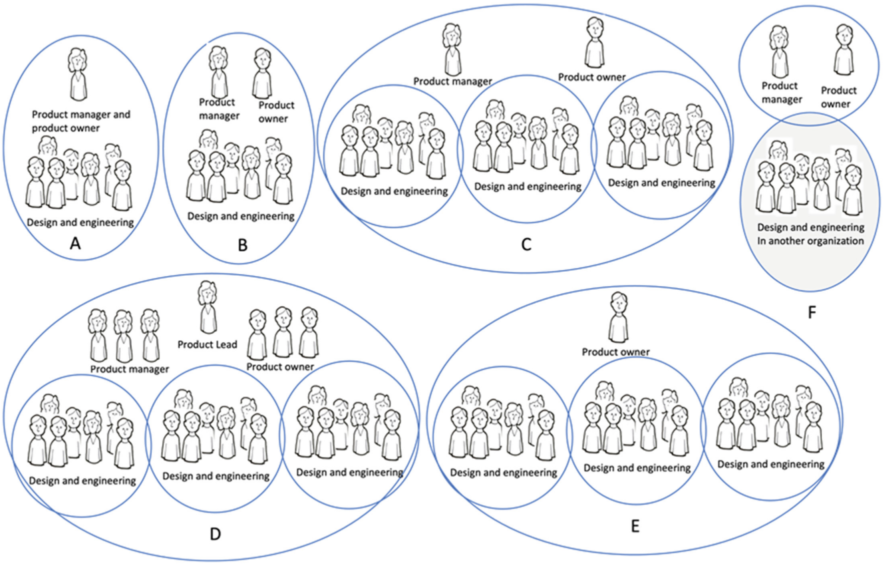


Fig. 1. Six SPM configurations with teams and SPM roles.

the same role as I have now before I came in, but he has very broad and long experience. He is a person I can go to and discuss with if there is anything I wonder about".

Configuration C: One PM, One PO, Several Product Teams

In this configuration, one PM and one PO share responsibility for two or more teams. There were several variants of this configuration depending on the product, as well as the different business units. One PO explained his situation: *"I work as the PO for three teams in our product and I am primarily responsible for the [removed for anonymity] solutions for DNB. … I don't think it's ideal for a PO to run three teams. That's the challenging part. Some days it's very difficult to work on anything productive apart from being a chat support person."* He continued: "*I have a PM whom I'm closely working with. … And I am more responsible for short-term [initiatives], like the next four months. That's my cup of tea, I would say. When it comes to what is going to be delivered this year, I expect a PM to plan that"* [I17].

A PO who worked with loans and credit explained that he made most of the product decisions: *"I also drive it as a PM as well…. If I need to get some clarifications or maybe push it, my escalation point for something else would be the PM. But apart from that, it's almost single-handedly driven"* [I15]. Further, he explained that the two development teams he was responsible for were in other business areas: *"They [IT department] have the Amazon Web Services developers, they have the UX designers, they have the mobile developers, they have the server-side developers, these are IT terms. They don't report to me, but they are part of the team that I am a PO for. Because there's no IT or tech resources or capability in my division."* He continued: "*Once I give them a priority, they know what to pick up*".

Configuration D: Several PMs, POs, and Teams, With a Product Lead

When there were several PM, PO, and teams, we found an additional SPM role, often referred to as the product lead. One product lead explained that he started as a PM with one team, but when they scaled from 15 to 60 people, they needed a new structure [I14]: "*I am responsible for the product managers, who in turn are responsible for product owners.*" The group of nine PMs, POs, and the product leads all had different roles and tasks, but they had shared responsibility for the yearly roadmap. He continued: "*Whether it is a product owner who helps prepare that year, or whether it is a product lead who prepares that year, does not matter that much. The most important thing is that everyone has ownership of it*" [I14]. Every quarter, the group updated the roadmap and the high-level priorities, which then were presented at a meeting with all 60 people. After this meeting, the discussions continued in the teams: *"The product owner primarily facilitated discussions in his own team"* [I14].

A similar setup was found in a product area with 13 teams in three sub-areas. One PM reported that in his sub-area he had three POs (out of eight in the whole area), each working in one team with a specific responsibility: "*Two teams are working on moving the existing solution up into the cloud. ... so, it is only one of the three teams that I can work with on new development*" [I06]. The job of migrating products to the cloud had a big priority in the whole organization as reported in the context description.

Configuration E: No PM, One PO, Several Teams

A similar configuration as the one described in Configuration A (except that there was no PM) was found in a product being migrated into the cloud (AWS). The customers would get a new customer experience, but no new functionality, which meant that there was less need for a PM at this stage, as the work was clearly defined and there was no need for strategy work. One PO who was responsible for three teams working on renewing a product described how there was to date no PM working with his teams: *"What I do is that I map today's functionality. And set the requirements for how the new functionality should be... And then I get to check it out with the businesspeople. The other thing I do is coordinate with UX. And the next step after that is to coordinate that we still have a value chain that works. So, we can go all the way from the customer down to the core system"* [I12].

Configuration F: One PM, One PO, the Team is External to the Organization

In one case, we found that although DNB set directions for the product, the product development was done off-site, by an external provider. "*We have a special structure compared to other DNB products, as we have a 'white label' service where we don't develop anything ourselves. So, I don't have any developers*" [I05]. In this set-up, there was "*a PM who handles the administrative [...] like compliance and vulnerability analyses and such while my main task is at the product level, the development progress and what we are to focus on in the time to come*" [I05]. Interestingly, DNB collaborated with other financial institutions in a product council, where they instead of acting like competitors together set directions and requirements for the development team.

Additional Configurations

We found four additional configurations. Because of page limitations, they are only listed: G) Several PMs, no POs, one team, H) No PM, one PO, one team, I) One PM, no PO, several teams, and J) Several PMs, one PO, several teams.

5 Discussion

The adoption of SPM in large-scale agile companies is growing, and companies like Google, Facebook, Amazon, and Microsoft use the practice. However, adopting SPM is challenging [19, 27], and how the practice is adopted is different from company to company. Therefore, there is a need for research on SPM in large-scale agile. We have described how these practices are applied in DNB, a fintech organization with 10,000 employees. In DNB a 'product' was defined *as something that delivers value, at the intersection of customer needs, business objectives, and technology enablement.* We will now answer our two research questions: *1) Which challenges do product management roles face in large-scale agile?* And 2*) What types of product management configurations exist in large-scale agile?*

5.1 SPM Challenges in Large-Scale Agile Organizations

First, we identified the top ten SPM challenges in large-scale agile in DNB (Table 2). The most common ones were challenges independent of a single product context, meaning that they were related to organizational features like company type and size. Further, we found that the organization was not set up to support products with long value chains that involved many complex subsystems where many organizational units needed to be involved (C4). Our findings are consistent with Maglyas' [19], who found that when product development involves several departments, flow is hindered because each unit acts independently and focuses on its own work instead of thinking about the whole product. Further, per Springer et al. [27], we found that there was a lack of alignment between organizational units (C10). Companies will have challenges with delivering products across departments when there are silos and missing alignment. These underlying problems are related to communication and synchronization. Springer et al. [27] argues that while PMs cannot change the company structure, they can still minimize its impact on the teams and product development process. One solution to these challenges is having regular meetings between POs and PMs [27]. We found such a solution in Configuration D, where a group of 9 PMs, POs, and a product lead, interacted regularly, in addition, the product lead and the PMs were responsible for working on aligning decisions across the organization.

Second, we found a lack of a unified way of working with goals (C8), which is similar to the challenge reported by Maglyas et al. [19] that there should be one way of tracing goals in SPM. Unclear goals is also a barrier to team autonomy [22]. Springer et al. [27] supports the importance of working on goals and argues that there is a need to identify and trace goals to be able to balance reactive (bugs, technical debt) and proactive work (new development). In Configuration D we found a sub-area with three teams that had clear goals: Two teams working on the cloud migration and one team working on new ideas.

Third, we found that there were many tasks with urgent priorities (C1), that prioritizations came from many different parts of the organization, and that many were unclear (C2). This finding is consistent with Springer et al. [27] who argues that when the strategy for products is changing frequently or when dependencies between products are unclear, PMs and their teams struggle with prioritization, seeing the long-term picture and being able to achieve outcomes, as the direction is changing often. Smite et al. [25] found similar problems at Spotify; when product teams have too many diverse tasks with urgent priority, individual goals become more important, joint goals become blurry, and the team's performance suffers. We have not had the opportunity to investigate why there were so many tasks with urgent priority and why the product strategy seems to shift frequently. However, Springer et al. [27] found that that if priorities change frequently, it is a signal that there might not be a strategy at all or the employers are simply not informed about the reasons behind priority changes, and how those decisions relate to product strategy. Our findings are also related to the problem of short-term thinking by Maglyas et al. [19]. But unlike the Maglyas et al., study from 2012, who argued that a one-year roadmap is too short, we did not find any problems with a one-year approach. This might be related to the fact that the market is more dynamic today than in 2012.

Fourth, we found that many product teams lacked a clear customer focus (C5), and put the business wants and needs first. Our findings are consistent with Maglyas et al. [19] that the decisions about new product development are often made internally in the company and not based on customer input. Fitting customer needs takes time and money. Springer et al. [27] argues that PMs have to run research and work iteratively to understand the customer needs, scaling opportunities, and customer willingness to pay for the product. The customer feedback loop is the key. In Configuration B, we describe a team with a data-driven culture [30] that focused on getting the product out before adjusting based on customer feedback, instead of doing customer research upfront.

5.2 SPM Configurations in Large-Scale Agile Organizations

Being able to succeed in delivering software frequently and iteratively requires work and knowledge coordination on different levels, i.e. at the portfolio, product, and team levels, and having access to the right resources. We found 10 different configurations of SPM (Fig. 1), and they were influenced by the following factors:

- The maturity and size of the team. For an autonomous, small, and mature product team, there was no need for a PO, as the team handled the PO activities themselves by translating customer needs and prioritizing the backlog.
- Product life cycle. In the startup phase, there was less need for a PO, and when migrating to the cloud (focus on technical issues) there was less need for a PM.
- The availability of supporting roles such as line managers and previous PMs. These supporting roles helped POs and PMs in conducting SPM.
- Where the team is located. If the team is located in a different department or in another organization, it might reduce interaction between PMs and the team, which again affects the SPM work.

We found that some configurations changed over time because of e.g., scaling (Configuration D), and some wanted to change as the setup was not optimal for the product development speed. These findings are consistent with Berntzen et al. [3], who found that scaling requires an adjustment in the coordination mechanism and in the organizational structure. Further, as resources are limited and prioritizations are sometimes conflicting, there is a need for SPM to negotiate [20] with other teams, business units and key stakeholders to ensure that the digital offerings make sense in terms of both users' needs and company revenue. Moreover, the six different configurations described in detail underscore that there are several ways of doing SPM. Still, there are key differences between SPM roles, where PM tasks are typically outward-focused and PO tasks are more inward-focused.

During the interviews, it became clear that the SPM roles were not performed in a unified manner in the organization. Sometimes a PM can be a PO and the other way around. Unclear roles and responsibilities can cause frustration and misunderstanding. One could argue that the roles should be standardized. However, as each product context was different, standardizations could also be a threat to efficiency. There is also no guaranteed how-to recipe to foster alignment on roles, and the question is thus how much standardization and autonomy should be built into the SPM setup and roles. Like the tale of the sitar player asking Buddha how best to tune his instrument, we find the famous answer, "Not too tight, and not too loose," as a good general rule on the degree of standardization of SPM roles.

5.3 Practical Implications

Based on our results, we can summarize some recommendations for those working with SPM. First, there does not exist one ideal SPM configuration. What constitutes an optimal configuration will vary over time, as the need for development resources of a product is not static. Further, as a product value chain changes over time, so does which part of the company will be involved in the product development. The larger the value chain, the more complex the SPM work and the more SPM resources are needed. Second, SPM should serve as a continuous link between customer needs, business needs and software development. Our results show that this is the essence of SPM regardless of the type and scale of the product, and the product life cycle. Third, independent of SPM configuration, the manager's social network, negotiation skills, and company knowledge are prerequisites for success. The consequence is that to become successful in SPM, there is a need to spend time establishing and maintaining the network.

Above all, managers of software products in large-scale agile organizations need to work data-driven, meaning that they need to be in control of user data and understand how to analyze such data from the customer interaction to improve and shorten the feedback loops.

6 Concluding Remarks and Future Research

Despite the increasing popularity of software product management in large-scale agile companies, little research exists on how SPM is performed in practice and what challenges and configurations exist. We have therefore conducted a case study to explore these challenges and configurations. Given today's increasing adoption rate of SPM in large-scale agile product companies, our findings can provide guidance for how SPM can work. The paper's main contributions are an overview of ten common SPM challenges in a large-scale agile fintech organization and a description of six SPM configurations. The results of our study constitute a step towards a practical and theoretical understanding of SPM. We found that the essence of the SPM roles in large-scale agile development is to make sure that the products are continuously linked with customer and business demand and that product development is data-driven. Setting up experiments and analyzing customer data is key for effective product development. Besides, sometimes a PM can fill the role of a PO and the other way around, meaning that what software product managers actually do is more important than the name of their role. Therefore, future research should study the tasks of software product managers, find solutions to the ten SPM challenges, and explore the relationship between the SPM configuration, product life cycle, and supporting roles. Further, to provide better guidance on SPM in large-scale agile, future research should investigate the advantages and disadvantages of the identified product management configurations, and how POs and PMs delineate their responsibilities and collaborate.

Acknowledgments. The authors thank the informants for their willingness to share their experiences. This research was supported by the Research Council of Norway through the TransformIT project (grant no. 321477).

References

1. Berntzen, M., et al.: A taxonomy of inter-team coordination mechanisms in large-scale agile. IEEE Trans. Software Eng. **49**(2), 699–718 (2022)
2. Berntzen, M., et al.: Coordination in agile product areas: a case study from a large FinTech organization. In: Agile Processes in Software Engineering and Extreme Programming (XP2024) (2024). (In Press)
3. Berntzen, M., et al.: Responding to change over time: a longitudinal case study on changes in coordination mechanisms in large-scale agile. Empir. Softw. Eng. **28**, 114 (2023). https://doi.org/10.1007/s10664-023-10349-0
4. Berntzen, M., et al.: The product owner in large-scale agile: an empirical study through the lens of relational coordination theory (2019)
5. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qual. Res. Psychol. **3**(2), 77–101 (2006). https://doi.org/10.1191/1478088706qp063oa
6. Conboy, K., Carroll, N.: Implementing large-scale agile frameworks: challenges and recommendations. IEEE Softw. **36**(2), 44–50 (2019)
7. Dingsøyr, T., et al.: Coordinating knowledge work in multiteam programs: findings from a large-scale agile development program. Proj. Manag. J. **49**(6), 64–77 (2018). https://doi.org/10.1177/8756972818798980
8. Dingsøyr, T., Fægri, T.E., Itkonen, J.: What is large in large-scale? a taxonomy of scale for agile software development. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (eds.) PROFES 2014. LNCS, vol. 8892, pp. 273–276. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13835-0_20
9. Dingsøyr, T., Moe, N.B.: Research challenges in large-scale agile software development. SIGSOFT Softw. Eng. Notes. **38**(5), 38–39 (2013). https://doi.org/10.1145/2507288.2507322

10. Ebert, C.: The impacts of software product management. J. Syst. Softw. **80**(6), 850–861 (2007). https://doi.org/10.1016/j.jss.2006.09.017
11. Ebert, C., Brinkkemper, S.: Software product management – an industry evaluation. J. Syst. Softw. **95**, 10–18 (2014). https://doi.org/10.1016/j.jss.2013.12.042
12. Fitzgerald, B., Stol, K.-J.: Continuous software engineering: a roadmap and agenda. J. Syst. Softw. **123**, 176–189 (2017)
13. Fricker, S.A.: Software product management. In: Maedche, A., Botzenhardt, A., Neer, L. (eds.) Software for People, pp. 53–81. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31371-4_4
14. Gustavsson, T., et al.: Changes to team autonomy in large-scale software development: a multiple case study of Scaled Agile Framework (SAFe) implementations. Int. J. Inf. Syst. Proj. Manag. **10**(1), 29–46 (2022)
15. Helferich, A., et al.: Product management for software product lines: an unsolved problem? Commun. ACM **49**(12), 66–67 (2006). https://doi.org/10.1145/1183236.1183268
16. Hyrynsalmi, S., et al.: A bibliographical study of software product management research. In: 2021 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), pp. 1–8 (2021). https://doi.org/10.1109/ICE/ITMC52061.2021.9570214
17. Larman, C., Vodde, B.: Large-Scale Scrum: More with LeSS. Pearson Education, Boston (2016)
18. Leffingwell, D.: SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises. Addison-Wesley Professional, Boston (2018)
19. Maglyas, A., et al.: Lean solutions to software product management problems. IEEE Softw. **29**(5), 40–46 (2012). https://doi.org/10.1109/MS.2012.108
20. Mikalsen, M., et al.: Agile digital transformation: a case study of interdependencies (2018)
21. Moe, N.B., et al.: Studying onboarding in distributed software teams: a case study and guidelines. In: Proceedings of the Evaluation and Assessment in Software Engineering, pp. 150–159 ACM, Trondheim (2020). https://doi.org/10.1145/3383219.3383235
22. Moe, N.B., Stray, V., Hoda, R.: Trends and updated research agenda for autonomous agile teams: a summary of the second international workshop at XP2019. In: Hoda, R. (ed.) XP 2019. LNBIP, vol. 364, pp. 13–19. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_2
23. Paasivaara, M., et al.: Large-scale agile transformation at Ericsson: a case study. Empir Softw. Eng. **23**(5), 2550–2596 (2018). https://doi.org/10.1007/s10664-017-9555-8
24. Skelton, M., Pais, M.: Team Topologies: organizing business and technology teams for fast flow. IT Revolution (2019)
25. Smite, D., et al.: Decentralized decision-making and scaled autonomy at Spotify. J. Syst. Softw. **200**, 111649 (2023). https://doi.org/10.1016/j.jss.2023.111649
26. Šmite, D., et al.: Software teams and their knowledge networks in large-scale software development. Inf. Softw. Technol. **86**, 71–86 (2017)
27. Springer, O., et al.: Strategies for dealing with software product management challenges. IEEE Access. **11**, 55797–55813 (2023). https://doi.org/10.1109/ACCESS.2023.3282605
28. Springer, O., Miler, J.: A comprehensive overview of software product management challenges. Empir. Softw. Eng. **27**(5), 106 (2022). https://doi.org/10.1007/s10664-022-10134-5
29. Stake, R.E.: Qualitative Research: Studying How Things Work (2010)
30. Storm, M., Borgman, H.P.: Understanding challenges and success factors in creating a data-driven culture (2020)

31. Tkalich, A., Ulfsnes, R., Moe, N.B.: Toward an agile product management: what do product managers do in agile companies? In: Stray, V., Stol, K.-J., Paasivaara, M., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming: 23rd International Conference on Agile Software Development, XP 2022, Copenhagen, Denmark, June 13–17, 2022, Proceedings, pp. 168–184. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-08169-9_11
32. Wagenblatt, Timo: Software Product Management: Finding the Right Balance for YourProduct Inc. Springer, Cham (2019)

Investigating Effort Estimation in a Large-Scale Agile ERP Transformation Program

Franziska Tobisch(✉), Karla Weigelt, Pascal Philipp, and Florian Matthes

TUM School of Computation, Information and Technology, Department of Computer Science, Technical University of Munich, Munich, Germany
{franziska.tobisch,karla.weigelt,pascal.philipp,florian.matthes}@tum.de

Abstract. Adaptability is vital in today's rapidly changing business environment, especially within IT. Agile methodologies have emerged to meet this demand and have thereby gained widespread adoption. While successful in smaller, co-located teams and low-criticality projects, applying agile methods in broader contexts poses challenges. Nevertheless, many organizations have started implementing agile methodologies in various areas, including large-scale Enterprise Resource Planning (ERP) projects. In contrast to traditional development, ERP projects involve deploying extensive integrated systems, are substantial in scale, and entail high risks and costs. Accurate predictions, like effort estimations, are crucial to meet customer satisfaction and deliver within plan and budget. However, estimating effort in an agile environment poses its own set of challenges. For instance, coordination efforts and dependencies among teams must be considered. While effort estimation is well-explored in classical software development and small-scale agile contexts, limited research exists in large-scale agile settings, particularly in projects rolling out and customizing standard ERP solutions. To address this gap, we conducted a case study on effort estimation in a large agile ERP transformation program, describing the estimation process, highlighting challenges, and proposing and evaluating mitigations.

Keywords: ERP systems · Large-scale agile · Effort estimation

1 Introduction

Today's unpredictable business environment requires organizations to be flexible and adaptable, especially in the IT sector [34]. As a result, many agile methodologies, e.g., Scrum, emerged [9]. The potential benefits of agile methods regarding faster delivery and customer satisfaction led to their wide adoption [7]. While agile practices proved successful in contexts characterized by small, co-located teams [8], limited system criticality, new developments, and frequent releases [23], applying them in other contexts without or only little adaption increases the risk of failure [17]. Still, the success on a small scale inspired many organizations to apply

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 70–86, 2024.
https://doi.org/10.1007/978-3-031-61154-4_5

agile methodologies outside of their initially intended context [7,8]. One example is Enterprise Resource Planning (ERP) rollout projects [16,20,35]. ERP systems are large-scale integrated systems covering most of a company's business processes [14]. In contrast to classical development projects, off-the-shelf ERP solutions by providers are rolled out and adapted to customer needs [14]. Those projects are often large [16] and involve high risk and costs [14,29].

One success factor in satisfying customers and delivering within plan and budget is accurate predictions like effort estimations [4,15,20]. However, estimating effort in an agile environment can be difficult [14]. Common techniques, like expert judgment and planning poker, are based on experts' opinions [28] and, thus, error-prone due to, e.g., human bias [15,30]. Also, constantly changing requirements complicate estimating accurately [27]. In scaled agile settings, estimating gets even more complex since, for example, coordination and dependencies of multiple teams [3,33] and the distribution of teams (e.g., increased communication effort) [17,32,33] become relevant. While effort estimation is well-researched in classical software development and small-scale agile contexts, little research has investigated this topic in large-scale agile settings [33]. In particular, how effort estimation is conducted in this context, potential challenges that can arise, and how they could be addressed have been barely investigated. Furthermore, settings that do not develop a new product but roll out and customize an existing ERP solution have yet to be considered. Thus, we conducted a case study investigating how effort is estimated within a large agile ERP transformation program. We describe the estimation process, present faced challenges, and make propositions to mitigate them. We defined the following research questions (RQs) to guide this study:

RQ1: How is effort estimation performed in a large-scale agile ERP program?
RQ2: What effort estimation challenges exist in a large-scale agile ERP program?
RQ3: How can the effort estimation challenges of such a program be addressed?

2 Background and Related Work

While ERP projects are classically organized with traditional approaches [14], the use of agile methods increases [16,20,35]. The differences between agile and traditional practices require new approaches to effort estimation [5]. Aligned to the incremental development in iterations, planning and effort estimation are done progressively and iteratively [5]. According to several studies investigating effort estimation in agile contexts [12,28], expert judgment, analogies, and planning poker are widely used techniques. The most commonly used unit for estimating is story points [12,27,31]. Next to studies investigating how effort estimation is conducted in agile software development, authors have studied existing challenges [13,21]. Those challenges include, for example, large project sizes and different understandings of requirements. In addition, Mallidi and Sharma [21] identified mitigation propositions to challenges in story point estimations, such as support by tools and Scrum Masters. Zia et al. [37] propose a framework to overcome challenges, and Tanveer et al. [30], who investigated multiple agile teams, focused on improving effort estimation, e.g., by tracking estimates.

Scaling agile practices adds complexity to effort estimation as, for instance, coordination and dependencies of multiple teams must be considered [3,33]. Several researchers have studied planning and effort estimation in large-scale and global agile software development to gain more insights. Usman and Britto [32] compared effort estimation in co-located and distributed agile software development. In both cases, expert judgment and story points are used most frequently, and effort is estimated mainly on iteration and release planning levels. The authors identified the distribution of teams as a cost driver, e.g., due to the required communication effort. Evbota et al. [11] investigated planning in a large-scale agile organization and identified challenges related to doing long-term estimates, requirement size, unclear requirements, and team commitment. Moreover, inefficient estimation events and unclear requirements were issues. Usman et al. [33] investigated effort estimation in a large-scale agile project. The authors found that the estimation is carried out in a two-stage process using expert judgment, estimating requirements more granularly in the second stage. Furthermore, the authors identified that the distribution of development teams and the requirement size and scope influence the estimation accuracy. Bick et al. [3] studied coordination challenges and misaligned planning in a development unit combining agile and traditional approaches. The effort is estimated on two levels: rough estimates on the inter-team level and highly granular estimates on the team level, which are adjusted if required. Core issues were a lack of dependency awareness, in-transparent estimates, and misalignment between top-down and bottom-up estimations. The authors propose, e.g., to hold cross-team planning workshops or unit-wide retrospectives. Finally, Kula et al. [15] investigated factors affecting on-time delivery in a scaled agile setting and highlighted, e.g., task and technical dependencies.

As presented, effort estimation in large-scale agile contexts was the subject of several studies. While Usman and Britto [32] focus on the impact of having distributed teams, other authors investigated effort estimation as part of related topics like coordination [3], planning [11] or on-time delivery [15]. Only Usman et al. [33] place a primary focus on effort estimation in large agile environments. Still, the authors [33] primarily investigate factors influencing the estimation accuracy instead of considering challenges in the effort estimation process and how practitioners can address those. Overall, empirical research providing detailed insights into estimation in scaled agile settings is still little [33], particularly on the potential challenges and related mitigations. Also, effort estimation in the context of large agile ERP projects, which, in contrast to the settings investigated so far [3,11,15,32,33], do not develop or maintain a product but roll out and customize an existing solution, has yet to be investigated.

3 Methodology

We conducted a holistic single-case study [36] and analyzed the collected data to answer our RQs. We chose this research methodology as case studies are a means to explore a *"contemporary phenomenon within its real-life context"* [36], like effort estimation in a scaled agile ERP roll-out program. To ensure a rigorous research design, we followed the guidelines of Runeson and Höst [25]. The case selection was intentional and aimed to identify a case typical for a large-scale agile ERP roll-out. The case program applies agile methods at scale as multiple agile teams work together on customizing and developing new functionalities for the ERP solution [10]. We conducted 16 interviews with 20 experts with various roles from all companies collaborating in the program (see Table 1).

Table 1. Interview partners

Interview No.	Alias	Company	Role	Large-Scale Agile Experience (years)
1	E1	SoftwareCo	Agile Coach, Scrum Master	6–10
2	E2	SoftwareCo	Project Manager	6–10
3	E3	SoftwareCo	Solution Architect	3–5
4	E4	EnergyCo	Developer	6–10
5	E5	EnergyCo	Product Owner	1–2
6	E6	EnergyCo	Product Owner	1–2
7	E7	SoftwareCo	Product Manager	3–5
8	E8	ConsultCo	Program Leadership support	6–10
9	E9	EnergyCo	Product Owner	6–10
10	E10	EnergyCo	Developer	3–5
11	E11	EnergyCo	Scrum Master	3–5
12	E12	EnergyCo	Business Contact	1–2
13	E13	SoftwareCo	Solution Architect	3–5
14	E14	ConsultCo	Program Leadership support	6–10
15	E15	SoftwareCo	Solution Architect	6–10
15	E16	EnergyCo	Product Manager, Product Owner	6–10
15	E17	EnergyCo	Business Process Expert	3–5
16	E18	EnergyCo	Solution Architect	6–10
16	E19	EnergyCo	Product Manager	3–5
16	E20	EnergyCo	Scrum Master	6–10

We conducted the case study between November and December 2022, mainly via semi-structured interviews. At the beginning of each interview, we ensured a shared understanding of relevant concepts (e.g., large-scale agile software development). Two researchers participated in each interview to enhance observer triangulation [25]. All interviews followed the same outline. First, we asked questions regarding the interviewees' experience and role within the program. Second, we asked questions exploring the program's effort estimation process. Third, we asked about the challenges experienced related to effort estimation within the program. The questions of the last two sections were open, allowing the interviewees to go into detail. We conducted all interviews using videoconferencing tools, recorded, and transcribed them. Next to the interviews, we included documents like presentations to facilitate the triangulation of data sources.

The collected data was analyzed and coded following the guidelines of Miles et al. [22] and Saldaña [26], applying a two-cycle approach and a combination of deductive and inductive coding. We resolved conflicts through discussion and mutual consent to increase the results' validity. In case of uncertainties regarding the data interpretation, we contacted the interviewees to resolve ambiguities.

4 Results

In the following, we present the results of our case study.

4.1 Context

The case study was conducted within a large transformation program at a German energy company (EnergyCo). After a merger, this program aims to standardize the ERP systems of the affected organizations by introducing a standard ERP cloud solution of a German software provider (SoftwareCo). Next to providing the ERP solution, SoftwareCo actively supports the program's leadership and implementation. In addition, a consulting company (ConsultCo) supports the program leadership. In total, 350 people are involved. The program applies an approach designed by SoftwareCo, supporting customers during the Preparation, Realization, Deployment, and Roll-out of the ERP solution. We focused on the program's Preparation and Realization Phase, in which agile practices are applied. The program's overall transformation time scope is five years (E8), including roll-outs in several hundred sub-companies of EnergyCo, each planned for 15 months. The program performs multiple roll-outs in parallel (E8, E9).

The program consists of the *Program Leadership*, Scrum teams grouped into *Workstreams*, and *Integration and Technology* teams. The *Program Leadership* is responsible for the program organization and coordination, progress monitoring, and steering. Implementation and customization tasks are performed by 20 Scrum teams, grouped into ten different *Workstreams*, each responsible for a business area (e.g., Reporting). Each Workstream has a designated Product Manager, acting as the link to the Program Leadership, providing assistance, tracking progress, and managing dependencies to other Workstreams. Each Scrum team consists of seven to twelve members: a Product Owner (PO), defining and prioritizing the Backlog and accepting work done, a Solution Architect, two to five team members responsible for technical design and implementation, and two to five team members responsible for documentation, functional specifications, and testing. A Scrum Master is coaching and supporting the team. Multiple cross-functional *Integration and Technology* teams support the Scrum teams in topics like training, change, or identity management.

4.2 Effort Estimation Process

This section describes the case program's effort estimation process (see Fig. 1) to answer our first RQ.

The Preparation Phase. This phase is performed at the end of a year to prepare for the Realization Phase in the upcoming year (E6, E9, E10, E16). Next to organizational preparations like staffing, the sub-companies, whose transformation is planned for the next year, gather Requirements, reflecting their custom needs for the ERP system based on the standard functionalities (E2, E7, E9, E10, E16, E18). The identified Requirements are then classified (E18), depending on whether they are part of the standard solution, require customization, must be developed ("gaps"), or are non-functional, are approved, and prioritized. According to E13, this classification influences the effort required as *"gaps [...] normally are related to real developments and not only to customizing or changing settings on the systems. So it's more to do for these objects and more to test and also to document."* An initial backlog is created (E16), and its items are assigned to the Workstreams. This backlog forms the basis for the following year (E6, E9, E10, E16) and the initial effort estimations in the Roadmap Planning (E5, E9, E19).

Roadmap Planning. This event happens once for each rollout sub-company (E1, E4, E5, E10, E11, E19). The effort for the prioritized Requirements from the initial backlog is estimated in person days (E1, E4, E13, E20). The persons involved in the planning event discuss the Requirements and agree on a rough estimate (E1, E2, E4, E6–8, E13, E16) based on experience and gut feeling. To simplify the estimation process, only a few people participate (E14), usually including the Product Manager(s), PO(s), Solution Architect(s), and some IT members of the responsible Workstream(s). The resulting Product Backlog, with prioritized and estimated Requirements, is the basis for the Realization Phase.

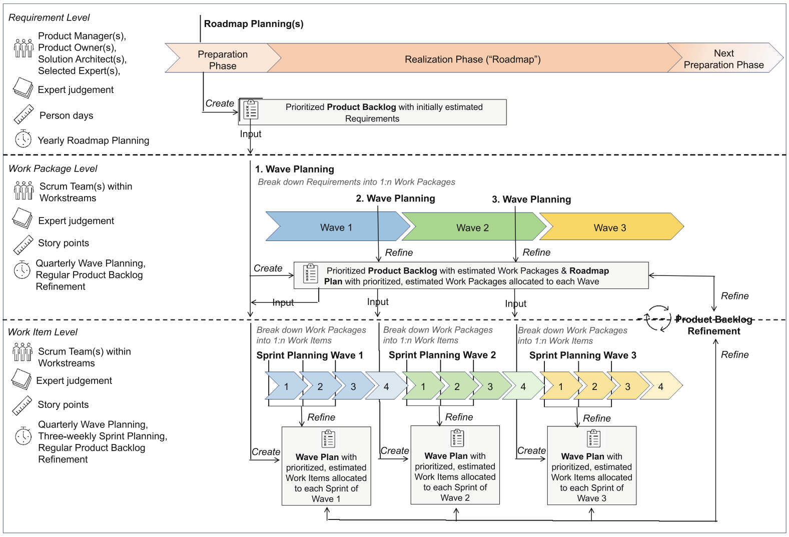

Requirement Level
Product Manager(s), Product Owner(s), Solution Architect(s), Selected Expert(s),
Expert judgement
Person days
Yearly Roadmap Planning
Roadmap Planning(s)
Preparation Phase
Realization Phase ("Roadmap")
Next Preparation Phase
Create
Prioritized Product Backlog with initially estimated Requirements
Input
Work Package Level
Scrum Team(s) within Workstreams
Expert judgement
Story points
Quarterly Wave Planning, Regular Product Backlog Refinement
1. Wave Planning
Break down Requirements into 1:n Work Packages
2. Wave Planning
3. Wave Planning
Wave 1
Wave 2
Wave 3
Refine
Refine
Create
Prioritized Product Backlog with estimated Work Packages & Roadmap Plan with prioritized, estimated Work Packages allocated to each Wave
Input
Input
Input
Refine
Product Backlog Refinement
Work Item Level
Scrum Team(s) within Workstreams
Expert judgement
Story points
Quarterly Wave Planning, Three-weekly Sprint Planning, Regular Product Backlog Refinement
Break down Work Packages into 1:n Work Items
Break down Work Packages into 1:n Work Items
Break down Work Packages into 1:n Work Items
Sprint Planning Wave 1
Sprint Planning Wave 2
Sprint Planning Wave 3
1
2
3
4
1
2
3
4
1
2
3
4
Refine
Refine
Refine
Refine
Create
Create
Create
Wave Plan with prioritized, estimated Work Items allocated to each Sprint of Wave 1
Wave Plan with prioritized, estimated Work Items allocated to each Sprint of Wave 2
Wave Plan with prioritized, estimated Work Items allocated to each Sprint of Wave 3

Fig. 1. Estimation process at the case program

The Realization Phase. This phase lasts nine months, split into three so-called Waves (E8). A Wave consists of three regular Sprints, in which Requirements are implemented, and a fourth one to test them and plan the next Wave (E11, E13). The effort is estimated in the *Wave Planning*, *Sprint Planning*, and *Product Backlog Refinement* by the Scrum teams (E1–20). POs are not actively estimating in these events but prioritize Backlog items, moderate the estimation, and support from a business perspective (E5–7, E9, E11, E15, E18). Estimations are done in normalized story points. One story point equals one person day (eight hours), and a "full" team member implementing Requirements has a capacity of 13 story points per Sprint. All teams use expert judgment to estimate.

Wave Planning. The first Wave Planning takes place before the Realization Phase officially begins. The second and third Wave Planning occur in the last Sprint of the first and second Wave. The basis for the first Wave Planning is the roughly estimated and prioritized Requirements in the Product Backlog. Each Requirement is broken down into n Work Packages (E1–20), functional subsets a team can implement in one Wave (E4), and estimated by the team responsible for it. Each Requirement's initial estimate is divided among the n Work Packages, based on their complexity, risk, and effort, by comparing them (E2, E5, E7). The teams adjust the Work Package estimates if the Requirement's estimate is too high or too low (E7, E14). Then, the Work Packages are allocated to the three Waves and ranked based on their priority in the so-called Roadmap Plan. Each Work Package must have a brief description, an effort estimation, and a dependency overview. In the second and third Wave Planning, the Roadmap Plan is refined. During each Wave Planning, the Work Packages of the upcoming Wave are further broken down into Work Items, complete and working functional subsets that a team can implement in one Sprint (E2, E4, E11). This breakdown into n Work Items and their estimation is performed the same way as the breakdown of Requirements (E1, E5, E7). The Work Item estimates, adjusted if the Work Package's effort was under- or overestimated, are the most accurate. The Work Items are then ranked based on priority and allocated to the three Sprints within the Wave, the so-called Wave Plan (E1, E2, E5, E7). In the following Sprint Plannings, the Wave Plan is refined. Depending on the Sprint allocation, each Work Item must have an initial effort estimation, dependency overview, and a brief or functional description. Each Wave Planning results in an updated Product Backlog, an updated Roadmap Plan, and a Wave Plan for the upcoming Wave. After each Wave Planning, each Scrum team's PO presents their planned Wave to all other POs to discuss dependencies and potential changes.

Sprint Planning. During this event, at the beginning of each Sprint, the whole Scrum team discusses and (re-)plans the Sprint (E1–20). The Work Items defined during the Wave Planning serve as a starting point, are discussed in detail, and then refined (e.g., ranking, allocated Sprint, estimation, description). The result of each Sprint Planning is an updated Sprint Backlog and Wave Plan.

Product Backlog Refinement. At least one six-hour refinement meeting is held in each Sprint. The teams adjust the estimations, prioritization, and breakdown into Work Packages or Items. Further, the teams discuss and estimate new Requirements (E3, E4, E7, E9, E12, E15, E19). This event ensures quick reaction to changes, that dependencies and wrong estimates are addressed, and that the Product Backlog is up-to-date (E3, E4, E7, E9, E12, E15, E19). Some teams have multiple, shorter refinements (E2, E11) or use their daily meetings (E20).

Tool Support. The primary tool used in the context of effort estimation within the program is designed by SoftwareCo and used as *"single source of truth"* (E2, E3, E5, E7, E9, E13, E15), e.g., for estimate and backlog documentation. Based on this tool, ConsultCo built an Excel reporting dashboard to visualize metrics and program progress (E1, E8, E14, E17). Excel is also used to document Requirements, including their estimation (E5–7, E9, E11, E16, E19, E20).

4.3 Effort Estimation Challenges

To answer our second RQ, we investigated the effort estimation challenges in the case program. We identified 14 challenges (C1–C14), which at least three interviewees mentioned [6] (see Table 2). This limitation is intended to ensure the criticality of the found challenges.

Table 2. Effort estimation challenges

Challenges	Experts
Project setting	
(C1) Project setting characterized by a fixed, large program time frame and tight budget restricts estimations, causing inaccurate initial estimates.	E2, E4, E10, E11, E13, E20
(C2) Time restrictions for estimating requirements contradict the time-consuming estimation process, making estimates inaccurate.	E2, E7, E9, E11–13, E16
(C3) Inappropriate tool support in the form of complexity, inflexibility, and a lack of supporting features make the requirement breakdown process, planning, and documentation of estimates difficult.	E7, E9, E13
(C4) Monitoring of estimates and actual effort is difficult due to the program's complexity and size, lacking transparency, and requirement size.	E1, E8, E14
Collaboration	
(C5) Lacking commitment to contribute to the estimation, e.g., due to resistance towards agile, estimations being seen as overhead, and lacking confidence in estimating, complicates estimating.	E1, E2, E6, E9, E11
(C6) Distribution of program leads to language barriers that hinder understanding requirements correctly and virtual meetings that complicate making estimates.	E1, E3, E5
(C7) Missing correct and common understanding of requirements, including quality criteria and scope, between individuals, teams, and workstreams complicates estimating.	E6, E7, E9, E10, E14
(C8) Considering dependencies to other teams, workstreams, and systems to estimate accurately is difficult, especially if unknown.	E2–5, E9, E13, E19
Expertise	
(C9) Unavailability of experts involved in the estimation process on a team level hinders it.	E3, E5, E7, E13
(C10) Lack of knowledge and experience of individuals regarding effort estimation make accurate estimations difficult.	E2, E5, E7, E12, E13
(C11) Neglection of relevant factors in initial estimations, e.g., dependencies, non-functional requirements, or the limited focus on requirement complexity during budget planning, hinder accurate estimation.	E1, E10, E13
Information deficit	
(C12) Information deficit in initial estimation of requirements, especially in the beginning, due to their size, potential dependencies, and uncertainty, makes estimating difficult.	E4, E6, E14
(C13) Unclear and incomplete specifications in functional descriptions of requirements make estimating difficult.	E1, E3, E5, E8, E11, E13–15, E18, E19
(C14) Unforeseen changes e.g., implementation problems, ad-hoc requirements, timeline changes, or personnel changes, complicate estimating.	E5, E9, E11, E12, E15, E17

We grouped the challenges into four categories: *Program setting*, *Collaboration*, *Expertise*, and *Information deficit*. Most challenges are related to the *Program setting* and *Collaboration*. The most frequently mentioned challenge is *Unclear and incomplete requirement specifications* (C13), which do not provide sufficient information about what has to be implemented and estimated and, thus, hinder accurate estimations of Work Packages and Items. E15 illustrates the challenge with an example: *"We are just handed over some sentences, [...] as an example, we need to have that button in blue. But we do not know where does that button now need to be positioned? What functionality does that button have?"*

4.4 Propositions to Mitigate Effort Estimation Challenges

To answer our third RQ, we reviewed academic literature and our interview data to identify mitigation propositions addressing the presented effort estimation challenges. In total, we found 19 propositions (M1–19), presented in Table 3.

Table 3. Mitigation propositions and addressed challenges

Mitigation Propositions	C1–14
(M1) Adding a buffer in case of uncertainty can help mitigate potential unknown efforts, upcoming changes, unclear or incomplete specifications (E14), and avoid too strict planning, especially for initial high-level estimations (E16).	C12–14
(M2) Pre-implementation phase to check requirements in detail, including their quality, can help enhance specifications (E3) and collect input for initial estimations. The pre-phase can also support considering relevant factors early on, e.g., dependencies, and establishing a shared, correct understanding of requirements, potentially reducing the time needed to estimate. Likewise, the literature recommends such a pre-phase with the customer [20].	C1, C2, C7, C8, C11–13
(M3) Maintaining a list of dependencies, e.g., between teams, facilitates considering them when estimating (E4, E19). Documenting each involved party's understanding of a requirement can also support a shared understanding (E10). Literature confirms the dependencies' relevance when estimating [3,30].	C7, C8
(M4) Deep dive sessions and workshops to clarify requirements and their implementation can help identify dependencies early, establish a correct, shared understanding (E5, E10, E16), collect information for requirements' initial estimations (E9), and enhance specifications (E5, E16). The knowledge gained can shorten the time required for estimating. Having the customer (i.e., the sub-companies) present in the meetings can help resolve ambiguities [31]. Tanveer et al. [30] argue that developing a shared understanding is often the most crucial. Overall, communication can greatly influence effort estimation [18].	C1, C2, C7, C12, C13
(M5) Early and continuous communication between all levels helps clarify uncertainties (E19), collect information (E4), and establish a correct, shared understanding of requirements. Teams should proactively seek help for estimating (e.g., experts (E5), training) or in case of language barriers (E5). Likewise, Lenarduzzi [18] highlights communication's impact on the effort estimation process. The exchange about dependencies makes affected parties aware of them (E5, E19) and supports their discovery [3]. Overall, regular and open communication can mitigate difficulties due to time and budget restrictions (E10).	C1, C6–10, C12, C13

(*continued*)

Table 3. (*continued*)

Mitigation Propositions	C1–14
(M6) Events to recap on learnings and document those, like Sprint Reviews, can improve next iterations. Such events can motivate and improve teams' commitment, support gaining knowledge and experience (E12) [11], reduce the impact of unavailable experts, and help consider relevant factors in the estimation in the future. Literature also recommends feedback sessions [1].	C5, C9–11
(M7) Guidelines and standard estimates for tasks uniform across all teams like documentation and testing can reduce the time required for estimating, do foster a shared understanding (E10) and do not require high knowledge and experience (E7), nor experts. Likewise, Tanveer et al. [30] propose to define standards.	C2, C7, C9, C10
(M8) Considering the effort of organizational and process factors and not only directly associated with the implementation can support estimating accurately. Considering who must be involved and, thus, communicate (E14) can help consider all relevant factors, like dependencies. Research [1,30,31] recommends considering aspects that affect the estimation, like developers' experience, optimistic estimators, and all activities [31], e.g., testing [27].	C8, C11
(M9) Involving experts with experience in estimating effort and the ability to react to unforeseen in the estimation process can help inexperienced teams estimate, learn it (E12, E13), and motivate them. Experts also have the experience to react to unforeseen changes (E12) and speed up the estimation (E7). Moreover, involving experts can foster a correct, shared understanding of the estimated requirements (E4, E9) and support management in initial estimations (E12). Likewise, the literature highlights the relevance of experts [28] and, for example, suggests actively involving Scrum Masters, fostering, e.g., appropriate estimation techniques [21].	C2, C5, C7, C9–11, C14
(M10) Normalizing story points to person days can help ensure everyone has a reference to the unit, giving the teams more confidence in estimating and increasing their commitment (E2). Normalization can reduce the need for respective knowledge or experts. Generally, a clear and standardized definition of story points within the project should be ensured [13].	C5, C9, C10
(M11) Planning requirements for an appropriate time frame can increase the limited estimation accuracy of requirements caused by the program setting (E19). More details about requirements can be available later (E9). Research confirms the difficulty of estimating requirements for large time frames [21,33] but recommends sufficient lead time to identify, e.g., dependencies, early [3].	C1, C8, C11, C12
(M12) Platforms and meetings to identify dependencies can increase the estimation accuracy. Regular exchange between Workstreams and teams can help avoid neglecting relevant factors, like dependencies, when estimating (E2–4, E13). Likewise, literature proposes cross-team workshops or retrospectives [3].	C8, C11
(M13) Support and motivation by Scrum Masters and Agile Coaches during the estimation process can help increase understanding, motivation, and acceptance of working and estimating in an agile way (E2) and reduce the impact of unavailable experts. Efforts by these roles can also help improve the quality of requirements' functional descriptions (E8). Likewise, Mallidi and Sharma [21] recommend support by Scrum Masters, e.g., to foster empowerment and self-organization.	C5, C9, C10, C13, C14

(*continued*)

Table 3. (*continued*)

Mitigation Propositions	C1–14
(M14) Automation enhances tool support (E7), reduces complexity and time required for estimating, and increases transparency, which makes monitoring estimations and efforts easier (E1, E8). Automation, e.g., initial estimates or analogies, as suggested by literature [1,15,30], can reduce the dependence on experts, knowledge, experience, and resistance. Enhanced tools could discover dependencies automatically (E3). Automation tools should be easy to use [30]. To prevent teams from not actively discussing, initial estimates can be used as a stimulus only [30].	C2–5, C8–10
(M15) Tool support like virtual whiteboards (E7) ease estimating, reducing the time needed and making it attractive for teams. Such tools ease collaboration despite local distribution, and documenting estimates supports monitoring. Collaboration and visualizations via tools can foster a shared understanding and help discover and record dependencies [11]. Literature agrees that features that visualize, e.g., proposed estimates, can support the estimation [30].	C2–8, C11
(M16) Tracking of actual efforts needed to implement requirements can help improve future estimates and understand reasons for delays (E7, E8). In addition, such tracking is the basis for monitoring. Literature confirms that accuracy metrics, e.g., actual effort, can improve the overall planning process [30].	C4
(M17) Supporting estimation techniques like Planning Poker or Silent Grouping [24] can simplify estimating (E4), reduce resistance and required time, foster discussions, and contribute to a shared, correct, and complete understanding of requirements (E4, E14). Increased understanding mitigates experts' unavailability. In distributed settings, such techniques can help overcome language barriers. The technique should be chosen based on project size and teams' experience [21].	C2, C5–7, C9, C10
(M18) T-shirt sizes as an estimation unit can simplify the estimation, ensuring everyone has a reference and understands the unit (E3). Improved understanding and a simplified process lower the need for experts and knowledge and reduce the time needed. T-shirt sizes also exclude controlling teams on the story point level, potentially increasing their commitment (E3). Also, literature [13] proposes using well-defined, objective size measurements.	C2, C5, C9, C10
(M19) Agile metrics and keeping a record of estimates can help to track estimates, and improving them can motivate teams. Learning from past estimates and metrics, e.g., velocity, can reduce the need for experts and foster learning. This proposition is also recommended by literature [21,30].	C4, C5, C9, C10

4.5 Evaluation of the Proposed Mitigations

Our evaluation aimed to assess to which degree practitioners agree with our propositions and collect qualitative insights into their opinions. Twelve experts who participated in our initial interviews evaluated our proposed mitigations in three semi-structured interviews and via a survey. We asked the participants to which degree they agreed with each proposition using a five-point Likert scale [19] and to provide qualitative feedback. We coded the qualitative data consistent with the coding in our case study. Figure 2 illustrates the evaluation results.

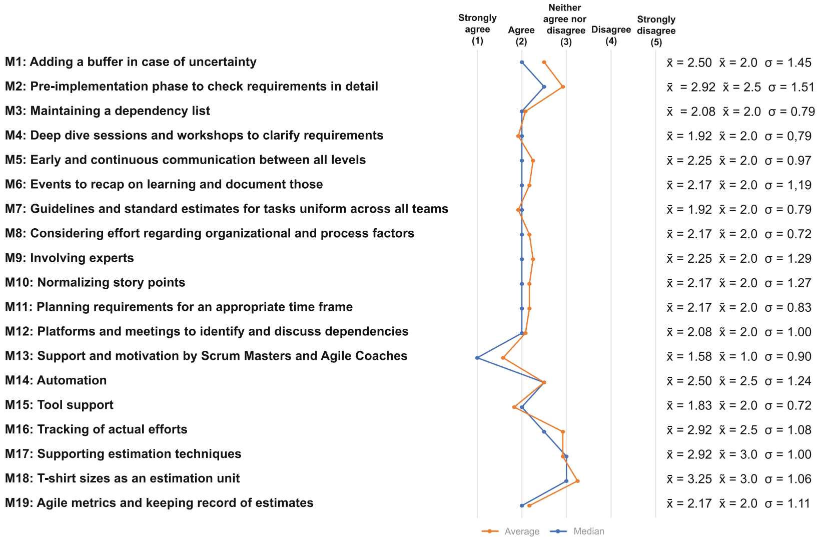


Fig. 2. Evaluation results of the mitigation propositions

The experts agreed with most propositions (M1, M3–15, M19). M13 received the highest agreement, followed by M15. The evaluation results of M1 and M2 diverged. Despite most experts agreeing with *Adding a buffer in case of uncertainty* (M1), some disagreed, expressing concerns, e.g., regarding their lack of traceability. Also, the opinions regarding a *Pre-implementation phase to check requirements in detail* (M2) diverged. While most experts think such a phase is valuable, several disagree. One expert claims that "*if you spend too much time over talking about design and not going forward, then you can also lose a lot of time.*" M18, the use of *T-shirt sizes as an estimation unit*, received the lowest agreement. While some respondents see T-shirt sizes as useful, e.g., pointing out the potential to simplify the estimation process for team members and the potential usefulness in an early phase, i.e., for higher-level estimates, others have a different opinion. These experts criticise, e.g., the imprecise nature and the additional work required to relate the estimates to budget resources.

5 Discussion

Next, we answer our RQs by discussing our study's key findings and limitations.

5.1 Key Findings

To shed light on how scaled-agile ERP programs estimate effort (RQ1), we investigated the effort estimation process within our case program. The program estimates effort at three levels with increasing granularity and decreasing time scope. A selected group performs the initial, rough estimation of high-level requirements, building the basis for the roadmap and following estimations. The Scrum teams are responsible for the more accurate estimations of the more granular functional sub-sets of the requirements, with medium to short-term time scopes. Also, Usman et al. [33] and Bick et al. [3] report on estimation processes in scaled agile settings with two phases building on each other. Large requirements are estimated roughly by a selected group [3], and teams estimate highly granular requirements more accurately during Sprint Plannings [3,33]. Our case program estimates low-level requirements iteratively and refines estimates regularly. Likewise, Bick et al. [3] found that the teams adjust estimates if required. Compared to classical ERP roll-outs [14], our case program estimates effort primarily in story points, typical for agile contexts [12,21,27]. Like in other small [27] and large agile settings [32,33], expert judgment is used to estimate. Overall, as described by Bick et al. [3], our case program's effort estimation approach combines long-term with agile aspects.

To answer RQ2, which effort estimation challenges exist in a large agile ERP program, we investigated the challenges our case program had to deal with concerning effort estimation. Our case program struggled with several challenges (C1, C2, C5, C6, C8, C9, C11-14), which were already reported as challenging or as factors that negatively influence estimation accuracy in other agile (e.g., [21,27,30,31]) and large-scale agile settings [3,11,15,32,33]. In line with this finding, Sandeep et al. [27] confirm that effort estimation in agile settings is challenging and Usman et al. [33] found that project scale complicates it further. The main challenge in our case program is unclear or incomplete requirement specifications (C13), hindering accurate estimation of the required effort, which multiple authors report [11,27,31,33]. In addition, we identified challenges, which, to the best of our knowledge, are barely or only partly reflected in existing studies of (scaled) agile settings (C3, C4, C7, C10). For example, we found a lack of knowledge and experience in estimating (C10) to be a challenge. Tanveer et al. [30], e.g., only report estimation experience being considered when estimating, Mallidi and Sharma [21] highlight its relevance, and Evbota et al. [11] mention that the practitioners in their scaled agile study had lacking expertise in estimating.

For answering our third RQ, we reviewed related literature and our interviewee data to identify mitigation propositions to address the found effort estimation challenges. Many of our propositions perceived as valuable align with agile values and principles [2] as they foster communication and collaboration (e.g., M4), transparency (e.g., M14), and continuous improvement and learning (M6). In general, open and early communication [35] and continuous feedback [20] are success factors for agile ERP projects. Having exchange opportunities (e.g., M4 & M5) and sufficient tool support (M15) can increase time efficiency (C2), help identify dependencies (C8), counteract the restrictive program setting (C1), and, ultimately, increase estimation accuracy. These insights align with other researchers' findings that tools can positively influence estimation accuracy [30] and that cross-level exchange is helpful regarding dependencies [3] in scaled agile settings. Our results show that, in particular, Scrum Masters and Agile Coaches can support the estimation process by assisting and motivating teams (M13). These findings reinforce Mallidi and Sharma [21], who highly recommend involving Scrum Masters, and Usman et al. [31], who claim that lacking the guidance of Scrum Masters can affect estimation accuracy negatively.

5.2 Limitations

We applied the assessment scheme of Runeson and Höst [25] to address the following potential validity threats. To mitigate threats to construct validity, we collected data from multiple sources (e.g., documents next to the interviews) to achieve data triangulation, gathered insights from interviewees with diverse roles and experiences, and clarified any ambiguity with the interviewees. Data triangulation also helped to address possible threats to internal validity. To mitigate threats to reliability, three researchers designed the interview questions to minimize reliance on any individual. Moreover, we made the data analysis and coding transparent by conducting them aligned to guidelines [22,26] and describing them. In regard to external validity, our findings are specific to the case program. However, the comprehensive description of our case program and the estimation process allows for an understanding of how, e.g., the identified challenges, relate to this context and may also be relevant in other programs.

6 Conclusion and Future Work

Our research was motivated by a need for more empirical studies on effort estimation in scaled agile contexts, highlighting existing challenges and approaches to mitigate those. In particular, settings that do not develop a new product but roll out and customize a standard ERP solution have yet to be considered. To make a first step towards filling this gap, we conducted a holistic single-case study at a large agile ERP transformation program to identify how effort is estimated in such programs. We presented identified challenges, proposed how to mitigate them, and evaluated those propositions. The case program combines long-term with agile aspects. The effort estimation takes place on different granularity levels, starting with high-level requirements on the roadmap level, broken down into functional subsets on a medium-time scope and Sprint level. A requirement's initial estimate is the basis for the estimations on a more granular level. Adjustments are made in case of over- or underestimations. Overall, the estimation accuracy increases with each breakdown step. The main estimation unit is story points but normalized to person days. While the requirements are initially estimated by a selected group, Scrum teams do the subsequent estimations. We found multiple effort estimation challenges common for agile and large-scale agile settings. The most mentioned effort estimation challenge is unclear and incomplete requirement specifications (C13). Overall, the evaluation participants agreed with most of our mitigation propositions. In particular, support from Scrum Masters and Agile Coaches (M13) was seen as valuable in counteracting existing challenges. Future research could validate our findings and test their applicability in other large agile ERP projects, scaled agile projects developing new products, or other agile organizations. Additional evaluations with practitioners could confirm the identified challenges. The reported mitigation propositions could be applied in practice to confirm their effectiveness.

Acknowledgments. This research is part of a bigger research project between TUM and SAP SE under the SAP@TUM Collaboration Lab and has been supported by SAP SE funding. The authors thank Thomas Röder for his support.

References

1. Basten, D., Sunyaev, A.: Guidelines for software development effort estimation. Computer **44**(10), 88–90 (2011)
2. Beck, K., et al.: Manifesto for Agile Software Development (2001). https://agilemanifesto.org. Accessed 12 Apr 2024
3. Bick, S., Spohrer, K., Hoda, R., Scheerer, A., Heinzl, A.: Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. IEEE Trans. Softw. Eng. **44**(10), 932–950 (2017)
4. Chow, T., Cao, D.B.: A survey study of critical success factors in agile software projects. J. Syst. Softw. **81**(6), 961–971 (2008)
5. Cohn, M.: Agile Estimating and Planning. Pearson Education, London (2006)
6. Coplien, J.: Software Patterns: Management Briefs. Cambridge University Press, Cambridge (1996)
7. Digital.ai: 16th Annual State of Agile Report (2022). https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf. Accessed 12 Apr 2024
8. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. J. Syst. Softw. **119**, 87–108 (2016)
9. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B.: A decade of agile methodologies: towards explaining agile software development. J. Syst. Softw. **85**(6), 1213–1221 (2012)
10. Dingsøyr, T., Moe, N.B.: Towards principles of large-scale agile development. In: Dingsøyr, T., Moe, N.B., Tonelli, R., Counsell, S., Gencel, C., Petersen, K. (eds.) XP 2014. LNBIP, vol. 199, pp. 1–8. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14358-3_1
11. Evbota, F., Knauss, E., Sandberg, A.: Scaling up the planning game: collaboration challenges in large-scale agile product development. In: Sharp, H., Hall, T. (eds.) XP 2016. LNBIP, vol. 251, pp. 28–38. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33515-5_3
12. Fernández-Diego, M., Méndez, E.R., González-Ladrón-De-Guevara, F., Abrahão, S., Insfran, E.: An update on effort estimation in agile software development: a systematic literature review. IEEE Access **8**, 166768–166800 (2020)
13. Hacaloglu, T., Demirörs, O.: Challenges of using software size in agile software development: a systematic literature review. In: Proceedings of the Joint Conference of the International Workshop on Software Measurement and the Conference on Software Process and Product Measurement 2018, pp. 109–122 (2018)

14. Ömüral, N.K., Demirörs, O,.: Effort estimation for ERP projects - a systematic review. In: Proceedings of the Euromicro Conference on Software Engineering and Advanced Applications 2017, pp. 96–103. IEEE, New York (2017)
15. Kula, E., Greuter, E., Van Deursen, A., Gousios, G.: Factors affecting on-time delivery in large-scale agile software development. IEEE Trans. Softw. Eng. **48**(9), 3573–3592 (2021)
16. Kraljić, A., Kraljić, T.: Agile software engineering practices in ERP implementation. In: Themistocleous, M., Papadaki, M. (eds.) Information Systems 2019. LNBIP, vol. 381, pp. 279–290. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44322-1_21
17. Kruchten, P.: Contextualizing agile software development. J. Softw. Evol. Process **25**(4), 351–361 (2013)
18. Lenarduzzi, V.: Could social factors influence the effort software estimation? In: Proceedings of the International Workshop on Social Software Engineering 2015, pp. 21–24. ACM, New York (2015)
19. Likert, R.: A technique for the measurement of attitudes. In: Archives of psychology (1932)
20. Madanian, S., Subasinghage, M., Tachiona, S.C.: Critical Success Factors of Agile ERP Development and Implementation Projects: A Systematic Literature Review. In: Proceedings of the Pacific Asia Conference on Information Systems 2021, 38, AIS (2021)
21. Mallidi, R.K., Sharma, M.: Study on agile story point estimation techniques and challenges. Int. J. Comput. App. **174**(13), 9–14 (2021)
22. Miles, M.B., Huberman, A.M., Saldaña, J.: Qualitative Data Analysis: A Methods Sourcebook, 4th edn. SAGE, Thousand Oaks (2019)
23. Nord, R.L., Ozkaya, I., Kruchten, P.: Agile in distress: architecture to the rescue. In: Dingsøyr, T., Moe, N.B., Tonelli, R., Counsell, S., Gencel, C., Petersen, K. (eds.) XP 2014. LNBIP, vol. 199, pp. 43–57. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14358-3_5
24. Power, K.: Using silent grouping to size user stories. In: Sillitti, A., Hazzan, O., Bache, E., Albaladejo, X. (eds.) Agile Processes in Software Engineering and Extreme Programming 2011. LNBIP, vol. 77, pp. 60–72. Springer, Cham (2011). https://doi.org/10.1007/978-3-642-20677-1_5
25. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**, 131–164 (2009)
26. Saldaña, J.: The Coding Manual for Qualitative Researchers, 4th edn. SAGE, Thousand Oaks (2021)
27. Sandeep, R.C., Sánchez-Gordón, M., Colomo-Palacios, R., Kristiansen, M.: Effort estimation in agile software development: a exploratory study of practitioners' perspective. In: Przybyłek, A., Jarzębowicz, A., Luković, I., Ng, Y.Y. (eds.) Lean and Agile Software Development 2022. LNBIP, vol. 438, pp. 136–149. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-94238-0_8
28. Schweighofer, T., Kline, A., Pavlic, L., Hericko, M.: How is effort estimated in agile software development projects? In: Proceedings of the Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications 2016, pp. 73–80 (2016)
29. Singh, A., Wesson, J.: Evaluation criteria for assessing the usability of ERP systems. In: Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists 2009, pp. 87–95. ACM, New York (2009)

30. Tanveer, B., Guzmán. L., Engel, U.M.: Understanding and improving effort estimation in agile software development: an industrial case study. In: Proceedings of the International Conference on Software and Systems Process 2016, pp. 41-50. ACM, New York (2016)
31. Usman, M., Mendes, E., and Börstler, J.: Effort estimation in agile software development: a survey on the state of the practice. In: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2015, pp. 1–10. ACM, New York (2015)
32. Usman, M. and Britto, R.: Effort estimation in co-located and globally distributed agile software development: a comparative study. In: Proceedings of the Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement 2016, pp. 219–224 (2016)
33. Usman, M., Britto, R., Damm, L.-O., Börstler, J.: Effort estimation in large-scale software development: an industrial case study. Inf. Softw. Technol. **99**, 21–40 (2018)
34. Van Oosterhout, M., Waarts, E., van Hillegersberg, J.: Change factors requiring agility and implications for IT. Eur. J. Inf. Syst. **15**, 132–145 (2006)
35. Wijaya, H.P.S.F., Kosala, R.R.: Development of an agile ERP framework for implementation: a systematic literature review. Int. J. Control Autom. **12**(5), 1–12 (2019)
36. Yin, R.K.: Case Study Research - Design and Methods, 4th edn. SAGE, Thousand Oaks (2009)
37. Zia, Z., Tipu, S., Zia, S.: An effort estimation model for agile software development. Adv. Comput. Sci. App. **2**(1), 314–324 (2012)

Value and Quality in Agile

The Current State of Operationalizing Value by Dutch Product Owners in Agile Software Development

Erik van Daalen[1](✉) and Rini van Solingen[2]

[1] KWD Resultaatmanagement, Edisonbaan 15, 3439 MN Nieuwegein, The Netherlands
erik.van.daalen@kwdrm.nl

[2] Delft University of Technology, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands
d.m.vansolingen@tudelft.nl

Abstract. Agile software development (ASD) teams show their relevance through the continuous delivery of valuable software. As part of ASD, product owners have an important role in four activities: determining most valuable backlog items, refining them, and in validating and measuring business value. In this research we investigate how product owners operationalize business value delivery in these activities. We conducted 38 semi-structured in-depth interviews with product owners in the Netherlands. Overall we found that 55% operationalize value using only two of the four activities, and that 11% apply all four activities. We furthermore found that only 26.5% of these product owners use a structured prioritization method, 47% prioritize based on individual preferences and gut feeling, and 26.5% do not use any prioritization approach at all. We found five different approaches to refinement. The majority (84%) of the product owners state that they validate business value delivery, while just 24% actually measure business value. Overall we conclude that there are opportunities to bring product ownership to a higher level on operationalizing business value delivery, and that there is a clear need in practice for better structure and guidance in operationalizing business value delivery to do so.

Keywords: Operationalizing business value delivery · prioritization product backlog · agile software development · refining business value · validating business value · measuring business value · product owner (PO) · agile practices

1 Introduction

Delivering business value by agile software development (ASD) teams is one of the core concepts within the agile domain [1]. The first principle of the agile manifesto is: '*Our highest priority is to satisfy the customer through early and continuous delivery of valuable software*' [2]. However: '*Satisfying the customer through continuous delivery of valuable software is not an easy task'* [3], says Jim Highsmith; one of the manifesto's authors.

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 89–106, 2024.
https://doi.org/10.1007/978-3-031-61154-4_6

Scrum is one of the most commonly used agile frameworks [1, 4, 5]. The fundamental unit of Scrum to deliver business value is a small-sized ASD team with a distinct division of roles. It consists of a scrum master, a product owner, and several developers [5]. The same is true for the agile teams mentioned in the most used agile scaling framework, SAFe [6]. Both frameworks prescribe that the product owner is accountable for maximizing the value delivered by the ASD team. Ultimately, in ASD the product owner is accountable for business value delivery to the stakeholders.

To maximize business value delivery, Scrum and SAFe contain a value delivery process [5, 6]. This process starts with putting requested value items on the product backlog. In most cases more details are required for the product owner to be able to prioritize these items. The ASD team also require details to make sure they deliver the requested value. These details are collected and added during the refinement by the product owner, in close collaboration with the stakeholders, and the ASD team. Based on the outcome of the refinement, the product owner can (re)prioritize the product backlog. The ASD team will take the highest item(s) from the product backlog to be delivered. Before, during, and after delivery, validation will take place with the stakeholders to determine if the requested value is delivered. Scrum and SAFe define the validation as a sprint and iteration review. A review is an informal acceptance of the solution potentially supported by measuring the value delivery with metrics [7].

The contribution of this research is to explore how product owners operationalize business value. Our main research question is: ***'How do product owners operationalize business value delivery with their agile software development teams?'*** Based on the process of value delivery, four sub-questions are identified:

RQ1: *How do product owners determine the most valuable backlog items?*
RQ2: *How do product owners refine business value?*
RQ3: *How do product owners validate business value delivery?*
RQ4: *How do product owners measure business value delivery?*

This paper is structured as follows: In Sect. 2, we present related work. Section 3 elaborates on our research method. The results of our research are presented in Sect. 4. In Sect. 5 we report our validity threats. Section 6 discusses our main findings and suggests future research opportunities. Section 7 summarizes our conclusions.

2 Related Work

In this section we present related literature on the four activities carried out by the product owner in the value delivery process [5, 6]: 1) prioritization, 2) refinement, 3) validation, and 4) measurement.

Prioritization

Because there are usually more requirements than feasible, given budget and schedule constraints, prioritization by the product owner is important to select the most valuable product backlog items [8, 9]. Scrum does not prescribe a prioritization method [5], therefore different methods are used. Hujainah et al. [9] identified 108 prioritization techniques. Jarzębowicz et al. [10] found that among the 69 Polish IT practitioners investigated, 4 prioritization methods are used. SAFe recommends a method called Cost of Delay (WSJF) [6]. Rodríguez et al. defined 47 value propositions used in value-based feature selection by interviewing 21 key stakeholders from 3 companies [11]. Other research shows that prioritization is only partly understood [8, 12, 13].

Refinement

A product backlog item needs to be refined before it can be prioritized, realized, and delivered [5, 6]. Business value is a broad concept that can be defined differently depending on the context and perspective, making refinement an essential step in agile. Not involving stakeholders will cause problems for the ASD team and lead to conflicts which can in turn lead to not delivering the expected solution and distraction from the final goal. Refinement is used within value-based software engineering (VBSE) [14–16], agile software development (ASD) [17, 18], and information system development (ISD) [12]. Biffl et al. [14] mention that benefits from value are not only monetary, but can also be economic, social, utilitarian, aesthetic, or ethical [14]. Salleh et al. [15] found in their systematic mapping study of 143 VBSE studies that there is no commonly accepted definition of value. 85% of VBSE studies refer to the value definition of Biffl et al. [14] and Boehm [16]: '*VBSE brings such value considerations to the foreground so that software engineering decisions at all levels can be optimized to meet or reconcile explicit objectives of the involved stakeholders, from marketing staff and business analysts to developers, architects, and quality experts, and from process and measurement experts to project managers and executives.'*

Chakraborty describes three important aspects of refinement: knowledge transfer, trust, and mental models/cognition [19]. Knowledge transfer is to overcome conflicting interpretations. This can be achieved by a continuous dialogue between the different stakeholders [19]. Trust can be increased by exchanging different points of view between stakeholders. The purpose of a shared mental model is to create a common understanding between product owner, stakeholders that request value, and the ASD team [19]. Deemer et al. stated that refinement contributes to preparing for the future and is part of continuous product development [20]. Palmer mentioned that the product owner has an explicit role in refinement [21]. Masood et al. [22] identified that refinement is done collectively with the entire team, but sometimes only between the product owner and the team lead.

Validation

It is important to check if the team is building the right product; this is called validation [7, 14, 15]. To make sure stakeholders get the requested value, it is important to validate before, during, and after realization. Within Scrum the sprint review is the validation event which is done amongst the stakeholders, product owner, and the ASD team [5]. Scrum does not prescribe how a sprint review is carried out, or which methods and techniques to use for review and validation [5].

Measurement

Research has been carried out on the formal measurement of business value delivery. Kupiainen et al. [23], Alahyari [18], Salleh et al. [24], and Sambinelli et al. [25] conclude that there is a lack of business value measurement methods and metrics within the area of agile software development. Kristinsdottir et al. [26], Dingsøyr et al. [1], and Huijgens et al. [27] conclude that measuring the actual business value delivered is difficult. Some researchers have made proposals of how to measure business value delivery; Hannay et al. define *benefit points* [28], and Hartmann et al. present *Net Present Value (NPV)*, *Internal Rate of Return (IRR)*, and *Return on Investment (ROI)* [29].

3 Research Method

To answer our research questions we conducted semi-structured in-depth interviews [30, 31] with 38 product owners in the Netherlands. The product owners can be differentiated in terms of their accountability for business, product, and technical decisions [32]. We used semi-structured interviews to collect data, uncover unexpected perspectives, elaborate on used practices, and collect insights on how product owners operationalize business value. We developed an interview protocol [31] to ensure consistency across the interviews.

The interview protocol contains four sections: 1) introduction, 2) opening question, 3) questions related to the four research questions, and 4) wrap-up. The introduction contains a very short explanation of the research in order to influence the participants as little as possible. During the introduction it was explicitly mentioned that the interviews are anonymous, and explicit approval was requested to record the interview. We kept our content-related questions to a minimum to allow the participants to give non-biased input as much as possible. We asked the participants the four research questions and requested examples, where available. The interview protocol was validated during the first interview and updated accordingly. In the wrap-up we asked the participants if they wanted to add anything that had not been touched upon. Furthermore we explained the follow-up steps after the interview. The interviews were conducted online (video and audio) via MS Teams, using the recording and automated transcription features. The interviews lasted between 33 and 81 min (average duration was 62 min). The interviews were conducted in Dutch. All transcriptions, coding, initial theme grouping, and determination of results was also done in Dutch, the native language of all participants and researchers. The translation of the results from Dutch to English was done while writing this paper. All interviews were fully transcribed, analyzed, and coded by the first researcher using reflective thematic analysis to determine patterns and construct themes [33]. The analysis was cross-checked by the second researcher. To minimize the risk of incorrect interpretations, a report was also created of each interview and sent to each participant for confirmation.

To address our research questions, we limited the participants to Dutch native-speaking product owners only. This was to ensure that we could exclude country-specific differences between the product owners. Speaking in the native language of the participants ensures that all nuances and details are covered, and limitations of participants expressing themselves in a non-native language can be excluded [34]. The validity consequences of this decision are discussed in Sect. 5.

We used a snowball sampling approach [35] in our personal network to ask connections to introduce us to 1, 2, or 3 product owners in their organization that we could interview. This resulted in 38 product owners from 17 organizations, across 10 different industries [36] (Table 1). The product owner experience of the participants can be found in Table 2.

Table 1. Overview of participating organizations and participants

Organization	Industry	Participants	Number of Employees	AEO (years)[1]	Agile methodology
O01	Travel	ID01; ID02; ID07	5,001–20,000	6–10	Scrum [5]; Kanban [37]; Lean Startup [38]
O02	Public	ID03; ID17; ID33	>20,001	6–10	Scrum [5]; Kanban [37]; Scrumban [39, 40]
O03	Insurance	ID04; ID06	2,001–5,000	6–10	Scrum [5]; Kanban [37]; Own Method
O04	Public	ID09; ID13; ID19	1,001–2,000	6–10	Scrum [5]; Kanban [37]
O05	Banking	ID08; ID10; ID14	>20,001	11–15	Scrum [5]; Kanban [37]; Own Method
O06	Manufacturing	ID05; ID16	5,001–20,000	3–5	Scrum [5]
O07	Manufacturing	ID15	2,001–5,000	6–10	Scrum [5]
O08	Public	ID18; ID20; ID27	501–1,000	3–5	Scrum [5]
O09	Travel	ID21; ID22; ID24	2,001–5,000	6–10	Scrum [5]; Kanban [37]; Lean Startup [38]
O10	Manufacturing	ID23	<500	11–15	Scrum [5]; Scrumban [39, 40]; Extreme programming (XP) [41]
O11	Education	ID35	2,001–5,000	1–2	Scrum [5]; Kanban [37]
O12	Public	ID12; ID25; ID26; ID32	1,001–2,000	6–10	Scrum [5]; Kanban [37]; Scrumban [39, 40]; Own Method; Extreme programming (XP) [41]
O13	Retail (Web shop)	ID11; ID28; ID29	2,001–5,000	6–10	Scrum [5]; Kanban [37]; Scrumban [39, 40]
O14	Computer programming	ID31	<500	6–10	Scrum [5]; Kanban [37]
O15	Consultancy	ID30; ID37	5,001–20,000	6–10	Scrum [5]; Kanban [37]

(*continued*)

Table 1. (*continued*)

Organization	Industry	Participants	Number of Employees	AEO (years)[1]	Agile methodology
O16	Travel	ID34	<500	3–5	Scrumban [39, 40]
O17	Agriculture	ID36; ID38	5,001–20,000	1–2	Scrum [5]; Kanban [37]

[1] AEO = agile experience of the organization

Table 2. Number of years of product owner experience

Number of years of experience as PO	#	%	Participants
Less than 1 year	0		–
1–2 years	12	32%	ID02; ID05; ID07; ID14; ID16; ID17; ID18; ID25; ID28; ID30; ID35; ID38
3–5 years	21	55%	ID03; ID04; ID06; ID08; ID09; ID10; ID11; ID12; ID19; ID20; ID21; ID23; ID24; ID26; ID27; ID29; ID31; ID32; ID33; ID34; ID37
6–10 years	5	13%	ID01; ID13; ID15; ID22; ID36

4 Results

This section presents the results of how product owners (POs) operationalize business value in practice. This section is structured according to the four research questions.

4.1 RQ1: How Do Product Owners Determine the Most Valuable Backlog Items?

In the interviews we found that 26.5% of product owners use a kind of structured method, 47% use individual prioritization, and 26.5% do not use any prioritization method at all (Table 3).

We determined two structured methods: Weighted Shortest Job First (WSJF) [6] and Desirability Viability Feasibility (DVF) [42].

Own structured methods are either variants of WSJF or methods that have been developed completely from scratch within the organization. An example provided by ID05 in the ‘Own structured method’ category was developed within the organization and uses five criteria to score an epic or feature. An epic and feature receive points for each criteria, see Table 4. The sum of all the five criteria determines the priority of an epic or feature (<10 is priority 5; $10 < \; =$ and < 15 is priority 4; $15 < \; =$ and < 20 is priority 3; $20 < \; =$ and < 30 is priority 2; > 30 is priority 1).

Table 3. Categorization of prioritization

Main category	Number of participants	%	Prioritization method used	Number of participants	%
Structured method	10	26.5%	Structured method	5	¬13%
			Own structured method	5	¬13%
Individual prioritization	18	47%	Individual list with criteria	5	¬13%
			Gut feeling	13	34%
Not using prioritization method	10	26.5%	Not using prioritization method	10	26.,5%

Structured method: ID02; ID06; ID13; ID21; ID24. Own structured method: ID01; ID05; ID30; ID32; ID37. Individual list with criteria: ID23; ID31; ID33; ID34; ID38. Gut feeling: ID03; ID10; ID12; ID15; ID16; ID17; ID18; ID19; ID22; ID25; ID27; ID28; ID29. Not using prioritization method: ID04; ID07; ID08; ID09; ID11; ID14; ID20; ID26; ID35; ID36.

Table 4. Practice provided by ID05: prioritization score table

Scoring criteria	Very low (0)	Low (3)	Medium (5)	High (7)	Top (10)
Financial value (recurring/year)	< 0.25 m€	0.25–0.5 m€	0.5–1.0 m€	1–2.5 m€	> 2.5 m€
Non-financial value	None	Limited	Significant	Substantial	Very high
Link to vision	No link	Limited	Part contribution	Substantial	Direct link, fully contributing to vision
Problem ownership	Problem is not recognized as priority at management level	Problem recognized as priority at management level, but resources are not available for the duration of the realization	Problem recognized as priority at management level, resources are identified but not yet available for the duration of the delivery	Problem recognized as priority at management level, resources are identified and available for the duration of the delivery	Problem recognized as top priority at management level, resources are identified and available for the duration of the delivery
Implementation	Hypothesis: levers will be difficult to implement and will require a long time	Hypothesis: levers will be difficult to implement but may be done fast if business environment changes	Hypothesis: levers will be implementable in the medium term	Hypothesis: levers will be easy to implement but require significant time	Hypothesis: levers will be easy to implement in a short time

For the five participants that use an individual list with criteria, we are not able to objectively determine any shared criteria.

4.2 RQ2: How Do Product Owners Refine Business Value?

Based on what the product owners mentioned, we determined five different refinement approaches. The main differences between these approaches is which stakeholders the product owner involves in refinement, and when the knowledge is transferred to the whole ASD team. Knowledge transfer is explaining the expected value by the stakeholders to the ASD team to create a shared mental model. Based on this we determine five different approaches:

1. Product owners refine with a sub-group of the ASD team.
2. Product owners refine with a sub-group of the ASD team and others.
3. Product owners refine on their own.
4. Product owners refine with a group of specialists that are not ASD team members.
5. Product owners refine directly with the whole ASD team.

We counted the number of times an approach was mentioned by the product owner, see Table 5. We conclude that 79% refine before the whole ASD team is involved. 21% directly involve the whole ASD team in the refinement. We also conclude that 65% of the product owners involve the whole or a few members of the ASD team in the refinement.

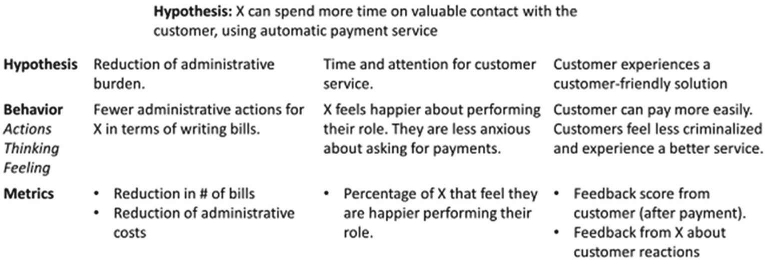
ID01 uses a value model to create a shared understanding of what the stakeholder requesting the value expects. This model consists of four components: 1) main hypothesis, 2) sub hypothesis, 3) behavior (actions, thinking, feeling), and 4) metrics. Figure 1

Table 5. Used refinement approaches

Refines business value before going to the whole ASD team	Number of participants	%	Refinement approaches used	Number of participants	%
Yes	30	79%	1. Sub-group of ASD team	11	29%
			2. Sub-group of ASD team and others	2	5%
			3. Product owner alone	8	21%
			4. Other group (no team members)	9	24%
No	8	21%	5. Together with whole ASD team	8	21%

1. Sub-group from ASD team: ID06; ID12; ID18; ID21; ID22; ID23; ID25; ID27; ID29; ID31; ID32. 2. Sub-group from ASD team and others: ID03; ID30. 3. Product owner alone: ID02; ID08; ID09; ID14; ID16; ID24; ID26; ID28. 4. Other group (no team members): ID05; ID11; ID15; ID17; ID34; ID35; ID36; ID37; ID38. 5. Together with whole ASD team: ID01; ID04; ID07; ID10; ID13; ID19; ID20; ID33.

contains an example. It starts with the hypothesis, which explains which stakeholder requires the business value and what is expected. The hypothesis still needs to be validated. It also describes the expected behaviors from the stakeholder that will use the delivered business value. It also defines how to measure the hypothesis.



Hypothesis: X can spend more time on valuable contact with the customer, using automatic payment service

Hypothesis	Reduction of administrative burden.	Time and attention for customer service.	Customer experiences a customer-friendly solution
Behavior *Actions* *Thinking* *Feeling*	Fewer administrative actions for X in terms of writing bills.	X feels happier about performing their role. They are less anxious about asking for payments.	Customer can pay more easily. Customers feel less criminalized and experience a better service.
Metrics	• Reduction in # of bills • Reduction of administrative costs	• Percentage of X that feel they are happier performing their role.	• Feedback score from customer (after payment). • Feedback from X about customer reactions

Fig. 1. Practical example of a value model

4.3 RQ3: How Do Product Owners Validate Business Value Delivery?

We found that the interviewed product owners use a variety of 17 different validation methods: panel of stakeholders (ID06, ID09, ID13, ID26, ID31, ID36, ID38), paid panel of stakeholders (ID25), technical validation (ID05, ID06, ID09, ID10, ID19, ID24, ID34), wireframes (ID06, ID25), employee surveys (ID09), automatic testing (ID12), prototyping (ID01, ID13), roadshows (ID13), user experience days (ID15), EPIC slides (ID17), mockups (ID20), AB testing (ID22, ID24), UX laboratory (ID22, ID24), guerilla testing (ID22, ID24), Google Analytics (ID26), integration testing (ID28), and customer visits (ID23).

We found that 84% of the product owners validate value and 16% do not validate value, see Table 6.

Table 6. Validate or not

Validates value	Number of participants	%	Participant IDs
Yes	32	84%	ID01; ID04; ID05; ID06; ID07; ID08; ID09; ID10; ID11; ID12; ID13; ID15; ID16; ID17; ID18; ID20; ID21; ID22; ID24; ID25; ID26; ID27; ID28; ID29; ID30; ID32; ID33; ID34; ID35; ID36; ID37; ID38
No	6	16%	ID02; ID03; ID14; ID19; ID23; ID31

One product owner mentioned using five different validation approaches (ID01):

1) They validate the hypothesis, determined when making the value concrete, via interviews with internal users and customers.
2) During the design sprint they validate the prototype (also mentioned by ID15).
3) The ASD team carries out a test with a few customers or internal users.
4) During realization they make use of an ambassador group of internal users.
5) They validate with customers after putting the solution into production.

4.4 RQ4: How Do Product Owners Measure Business Value Delivery?

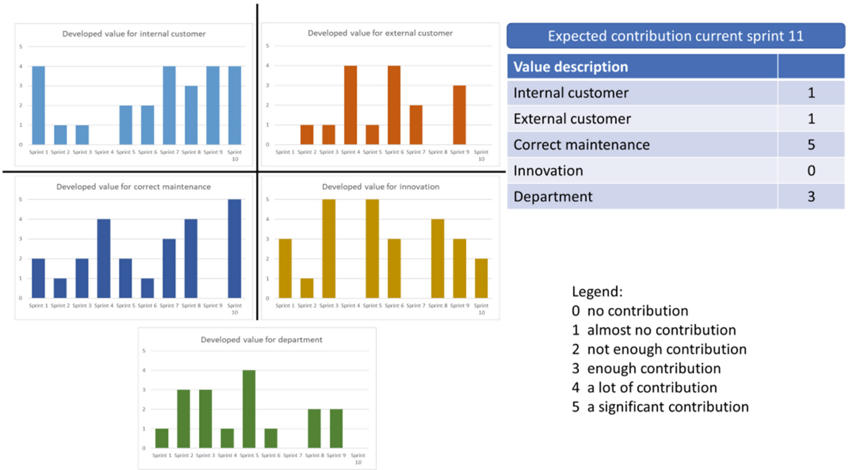
We found that 24% of the product owners state that they use metrics to measure business value, whereas 76% of the product owners state that they do not use metrics to measure value, see Table 7.

Table 7. Measuring business value

Using metrics to measure value	Number of participants	%	Participant IDs
Yes	9	24%	ID01; ID04; ID05; ID06; ID14; ID16; ID21; ID22; ID36
No	29	76%	ID02; ID03; ID07; ID08; ID09; ID10; ID11; ID12; ID13; ID15; ID17; ID18; ID19; ID20; ID23; ID24; ID25; ID26; ID27; ID28; ID29; ID30; ID31; ID32; ID33; ID34; ID35; ID37; ID38

We found that the 24% that use metrics measure these after value delivery. Some of them made estimates upfront during refinement and validation. The product owners that do use metrics to measure value mentioned some examples of how they measure business value delivery. The metrics used by ID01 are mentioned in Fig. 1 above.

ID25 developed a value monitor, see Fig. 2. The monitor contains five perspectives of value: internal customer, external customer, contribution to correct maintenance, innovation, and to the value the department delivers. The score is a scale from zero to five, whereby zero is no contribution and five is a significant contribution. This monitor is filled in by the product owner and the ASD team based on feedback from their stakeholders.



Expected contribution current sprint 11

Value description	
Internal customer	1
External customer	1
Correct maintenance	5
Innovation	0
Department	3

Legend:
0 no contribution
1 almost no contribution
2 not enough contribution
3 enough contribution
4 a lot of contribution
5 a significant contribution

Fig. 2. Practical example provided by ID25: value monitor

4.5 Main Question: How Do Product Owners Operationalize Business Value Delivery with Their Agile Software Development Teams?

In Table 8 we provide an overview of the number of activities used by the interviewed product owners. The four activities of operationalizing business value are: 1) determining most valuable backlog items, 2) refining expected business value, 3) validating business value, and 4) measuring delivered business value. We found that 4 of the 38 product

Table 8. Number of operationalizing business value activities used by the 38 product owners

Number of activities	Operationalizing business value				#	%	Participant IDs
	1) Use structured method	2) Refines	3) Validates	4) Measures			
4	✓	✓	✓	✓	4	10.5%	ID01; ID05; ID06; ID21
3	✓	✓	✓		9	24%	ID13; ID24; ID30; ID32; ID37
		✓	✓	✓			ID04; ID16; ID22; ID36
2		✓	✓		21	55%	ID11
	✓	✓					ID02
		✓	✓				ID07, ID08, ID09, ID10, ID12, ID15, ID17, ID18, ID20, ID25, ID26, ID27, ID28, ID29, ID33, ID34, ID35, ID38
		✓		✓			ID14
1		✓			4	10.5%	ID03; ID19; ID23; ID31
Percentages (yes)	26%	100%	84%	24%			

owners use all 4 activities to operationalize business value. The majority of the product owners (55%) use only 2 activities to operationalize business value. We found that all product owners refine business value. We found that 84% validate business value. We found that just 26% use a structured method to prioritize backlog items based on business value. We also found that just 24% of the product owners use metrics to measure the delivered business value.

We observed differences between organizations but also between different product owners in the same organization.

5 Validity Threats

To evaluate the quality of our research, we reflect on the threats to the four validity dimensions: construct validity, internal validity, external validity, and reliability [30, 31, 43]. We mainly followed the validity guidelines from Runeson and Höst [43].

Construct validity concerns to what extent the operational measures being studied truly represent what the researcher has in mind and what is investigated according to the research questions [43]. We created an interview protocol containing all questions and explanations given to the participant. The interview protocol ensured that the same questions were asked and a consistent explanation was given to the participants each time. We used the answers and examples provided by the participant faithfully. When a participant, for example, stated that they use Scrum, we did not further investigate or question to which extent Scrum [5] is correctly implemented.

Internal validity concerns causal relations between investigated factors. Our research explored how product owners operationalize business value and does not aim to associate relationships [43]. All four sub-questions were analyzed separately. No causal relationships between the sub-questions were analyzed.

External validity is concerned with the extent to which our conclusions are valid outside the population participating in this research [43]. The external validity of our research is limited. Firstly, an individual product owner does not represent the other product owners in that organization. We attempt to mitigate this by selecting product owners from different industries and agile experience of the organization. Secondly, the included organizations are not representative of other organizations (also not within the same industry). Thirdly, the interviews are only held with product owners in the Netherlands, and are therefore not representative of other countries in Europe or the world. This is confirmed by the research differences with Polish product owners on prioritization [10]. Finally, participants have been selected by approaching product owners through the personal network of the researchers. This resulted in 38 product owners from 17 organizations, within 10 different areas of industry.

Reliability refers to the extent to which data and the analysis are dependent on the specific researchers [43]. To overcome cultural differences [44] and misinterpretations [34] of meaning of words, we decided to narrow our research to organizations from the Netherlands and native Dutch-speaking product owners. All questions and interviews were held in Dutch. We created an interview protocol based on the research questions.

All interviews were conducted online (video and audio) via MS Teams, using the recording and automated transcription features of MS Teams. The first researcher conducted the reflective thematic analysis, which was cross-checked by the second researcher. To minimize the risk of incorrect interpretations, a report was also created of each interview and sent to each participant for confirmation. These approved interview reports were used to determine the results. All participants have confirmed the content of their report.

6 Discussing Research Questions and Future Research

RQ1: How Do Product Owners Determine the Most Valuable Backlog Items? We found three main categories of prioritization: structured method (26.5%), individual prioritization (47%), and not using any prioritization (26.5%). Kukreja et al. [45] carried out research in 2011 and concluded that various ad hoc prioritization techniques are used and that there is a need for structured methods. We argue that a structured prioritization method or framework will contribute to the effectiveness of delivering business value, confirmed by Hujainah et al. [9] and Jarzębowicz et al. [10] and Kantola et al. [46], for product owner teams. We argue that these low figures for using a structured method and only three methods can be explained by the fact that Scrum does not prescribe any prioritization method, and SAFe recommends only one method: WSJF [6]. We also conclude that methods identified by Hujainah et al. [9] do not appear to find their way to product owners.

We suggest carrying out further empirical research into which structured methods can support product owners to optimize business value delivery.

RQ2: How Do Product Owners Refine Business Value? We found that all 38 participants used refinement methods. The results further confirm the work of Palmer [21], where all product owners mentioned that they carry out refinement. Furthermore we have identified five different approaches to how product owners refine value. The advantage of approaches 1, 2, and 5 in relation to approaches 3 and 4 is that the understanding of business value by the ASD team members is likely to be higher, because the whole ASD team, or certain members thereof, are participating in building up the mental model, and in knowledge transfer from the stakeholders that request the business value. We found four approaches where 79% refine the business value *before* the whole ASD team gets involved. And one approach (21%) that directly involves the whole ASD team. This finding is interesting as it contradicts the agile approach, which stresses the importance that ASD team members are in direct contact with their stakeholders (fourth principle of the agile manifesto [2]).

Future research is required to determine what the pros and cons are of these approaches, and if these approaches can contribute to more involvement of business stakeholders as suggested by Alami et al. [47] and Hoda et al. [48].

RQ3: How Do Product Owners Validate Business Value Delivery? We found that there is no one common way of carrying out validation, and that a variety of different methods are used. Scrum [5] mentions the sprint review but does not prescribe how to perform this. Salleh et al. [15] identified nine studies in the VBSE principles & practices VB verification and validation. We argue that validation methods appear to be context-specific and that product owners should therefore have more knowledge of when and how to use which specific validation method.

The extent to which product owners are aware of the broad availability of such methods, their applications, strengths and weaknesses, and how product owners are trained to apply them appropriately, is a topic for future research.

RQ4: How Do Product Owners Measure Business Value Delivery? We see that just 24% of the product owners explicitly measure value. Our research confirms the conclusions of Kristinsdottir et al. [26], Dingsøyr et al. [1], and Huijgens et al. [27] that measuring delivered business value is difficult.

We suggest investigating if a measurement system can be developed for measuring business value delivery, taking into consideration, for example, the 47 value propositions mentioned by Rodríguez et al. [11].

We suggest research to investigate how measuring business value can contribute to increasing the delivery of business value, prove that the delivered value relates to the investment, and on how to select and use the most suitable metrics.

Main Research Question: How Do Product Owners Operationalize Business Value Delivery with Their Agile Software Development Teams? We found that only 10.5% of the product owners are using all four activities of operationalizing business value, 24% are using three activities, the majority (55%) are using only two activities, and 10.5% use just one activity. There could be an explanation for these low figures because the 'ideal product owner' does not exist. Kadenic et al. [32] come to the conclusion that the ideal product owner does not exist or that the responsibility of the product owner cannot be carried out by one person. Based on these findings and research sub-questions, we argue that operationalizing business value delivery by product owners in practice requires improvement. Organizations should invest in further development of the skills and capabilities of the product owner, which should contribute to increasing the level of business value delivery. This is also supported by the research of Kantola et al. [46].

7 Conclusion

The extent to which product owners operationalize business value delivery differs considerably in practice. These differences are not only observed between organizations but even between different product owners in the same organization. In order to improve business value delivery from the product owner perspective, there is a clear need for better structure and guidance in operationalizing business value delivery in daily practice.

This is also confirmed by the lack of structured methods or frameworks used by these 38 product owners. Some local practices have been found in the interviews, yet only a few product owners operationalized business value delivery in a repeatable and structured way.

Furthermore we conclude that, in practice, operationalizing business value delivery is not in place. Only four product owners are applying all four activities to operationalize business value delivery. We found large differences in perspectives among product owners between validating and measuring value. The majority (84%) of the product owners in this research do claim to validate value, yet measuring value is done only by a few of them (24%). A structured method to operationalize business value could as such bridge this gap between perceived validation and collecting quantitative evidence.

In general we conclude that there are several opportunities to take product ownership to the next level in operationalizing the delivery of business value in practice.

References

1. Dingsøyr, T., Lassenius, C.: Emerging themes in agile software development: introduction to the special section on continuous value delivery. Inf. Softw. Technol. **77**, 56–60 (2016). https://doi.org/10.1016/j.infsof.2016.04.018
2. Beck, K., Beedle, M., Bennekum, A.V., Cockburn, A.: The agile manifesto (2001). https://agilemanifesto.org/
3. Highsmith, J.A., Highsmith, J.: Agile Software Development Ecosystems. Addison-Wesley Professional, Boston (2002)
4. Digital.ai AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf
5. Sutherland, J., Schwaber, K.: The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game (2020)
6. SAFe 6.0 Framework. In: Scaled Agile Framework. https://scaledagileframework.com/. Accessed 5 Jun 2023
7. Balci, O.: Validation, verification, and testing techniques throughout the life cycle of a simulation study. Ann. Oper. Res. **53**, 121–173 (1994). https://doi.org/10.1007/BF02136828
8. Kukreja, N., Payyavula, S.S., Boehm, B., Padmanabhuni, S.: Value-based requirements prioritization: usage experiences. Procedia Comput. Sci. **16**, 806–813 (2013). https://doi.org/10.1016/j.procs.2013.01.084
9. Hujainah, F., Bakar, R.B.A., Abdulgabber, M.A., Zamli, K.Z.: Software requirements prioritisation: a systematic literature review on significance, stakeholders, techniques and challenges. IEEE Access **6**, 71497–71523 (2018). https://doi.org/10.1109/ACCESS.2018.2881755
10. Jarzębowicz, A., Sitko, N.: Communication and documentation practices in agile requirements engineering: a survey in polish software industry. In: Wrycza, S., Maślankowski, J. (eds.) SIGSAND/PLAIS 2019. LNBIP, vol. 359, pp. 147–158. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29608-7_12
11. Rodríguez, P., Urquhart, C., Mendes, E.: A theory of value for value-based feature selection in software engineering. IEEE Trans. Softw. Eng. **48**, 466–484 (2022). https://doi.org/10.1109/TSE.2020.2989666

12. Racheva, Z., Daneva, M., Sikkel, K., Buglione, L.: Business value is not only dollars – results from case study research on agile software projects. In: Ali Babar, M., Vierimaa, M., Oivo, M. (eds.) PROFES 2010. LNCS, vol. 6156, pp. 131–145. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13792-1_12
13. Bakalova, Z., Daneva, M., Herrmann, A., Wieringa, R.: Agile requirements prioritization: what happens in practice and what is described in literature. In: Berry, D., Franch, X. (eds.) REFSQ 2011. LNCS, vol. 6606, pp. 181–195. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19858-8_18
14. Biffl, S., Aurum, A., Boehm, B., et al.: Value-based software engineering. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-29263-2
15. Salleh, N., Mendes, E., Mendes, F., et al.: Value-based software engineering: a systematic mapping study (2022). https://dx.doi.org/10.2139/ssrn.4148149
16. Boehm, B.: Value-based software engineering. ACM SIGSOFT Softw. Eng. Notes **28**, 4 (2003). https://doi.org/10.1145/638750.638775
17. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B.: A decade of agile methodologies: towards explaining agile software development. J. Syst. Softw. **85**, 1213–1221 (2012). https://doi.org/10.1016/j.jss.2012.02.033
18. Alahyari, H., Svensson, R.B., Gorschek, T.: A study of value in agile software development organizations. J. Syst. Softw. **125**, 271–288 (2017). https://doi.org/10.1016/j.jss.2016.12.007
19. Chakraborty, S., Sarker, S., Sarker, S.: An exploration into the process of requirements elicitation: a grounded approach. J. Assoc. Inf. Syst. **11**, 1 (2010). https://doi.org/10.17705/1jais.00225
20. Deemer P, Benefield G, Larman C, Vodde B (2012) A lightweight guide to the theory and practice of scrum. https://scrumprimer.org/scrumprimer20_small.pdf
21. Palmer, K.: Product owner agile systems engineering strategies. INCOSE Int. Symp. **23**, 346–355 (2013). https://doi.org/10.1002/j.2334-5837.2013.tb03023.x
22. Masood, Z., Hoda, R., Blincoe, K.: Real world scrum a grounded theory of variations in practice. IEEE Trans. Softw. Eng. **48**, 1579–1591 (2022). https://doi.org/10.1109/TSE.2020.3025317
23. Kupiainen, E., Mäntylä, M.V., Itkonen, J.: Using metrics in agile and lean software development–a systematic literature review of industrial studies. Inf. Softw. Technol. **62**, 143–163 (2015). https://doi.org/10.1016/j.infsof.2015.02.005
24. Salleh, N., Mendes, F., Mendes, E.: A systematic mapping study of value-based software engineering. In: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 404–411. IEEE (2019). https://doi.org/10.1109/SEAA.2019.00067
25. Sambinelli, F., Borges, M.A.F.: The Strategies to increase customer value in agile: a survey of Brazilian software industry. J. Inf. Syst. Eng. Manag. **4**(2), em0090 (2019). https://doi.org/10.29333/jisem/5889
26. Kristinsdottir, S., Larusdottir, M., Cajander, Å.: Responsibilities and challenges of product owners at spotify - an exploratory case study. In: Bogdan, C., Gulliksen, Jan, Sauer, Stefan, Forbrig, Peter, Winckler, Marco, Johnson, Chris, Palanque, Philippe, Bernhaupt, Regina, Kis, Filip (eds.) HCSE/HESSD -2016. LNCS, vol. 9856, pp. 3–16. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44902-9_1
27. Huijgens, H., Van Deursen, A., Van Solingen, R.: The effects of perceived value and stakeholder satisfaction on software project impact. Inf. Softw. Technol. **89**, 19–36 (2017). https://doi.org/10.1016/j.infsof.2017.04.008

28. Hannay, J.E., Benestad, H.C., Strand, K.: Agile uncertainty assessment for benefit points and story points. IEEE Softw. **36**, 50–62 (2018). https://doi.org/10.1109/ms.2018.2875845
29. Hartmann D, Dymond R (2006) Appropriate agile measurement: using metrics and diagnostics to deliver business value. AGILE 2006 (AGILE'06). https://doi.org/10.1109/AGILE.2006.17
30. Yin, R.K.: Case Study Research and Applications. Sage, Thousands Oaks (2018)
31. Robson, C.: Real World Research: A Resource for Social Scientists and Practitioner-Researchers. Wiley-Blackwell, Hoboken (2002)
32. Kadenic, M.D., de Jesus Pacheco, D.A., Koumaditis, K., et al.: Investigating the role of product owner in scrum teams: differentiation between organisational and individual impacts and opportunities. J. Syst. Softw. **206**, 111841 (2023). https://doi.org/10.1016/j.jss.2023.111841
33. Braun, V., Clarke, V.: Thematic Analysis: A Practical Guide. SAGE, Los Angeles (2022)
34. Hofstadter, D.R., Sander, E.: Surfaces and Essences: Analogy as the Fuel and Fire of Thinking. Basic books, New York (2013)
35. Goodman, L.A.: Snowball sampling. Ann. Math. Stat. 148–170 (1961). https://doi.org/10.1214/aoms/1177705148
36. United Nations. International Standard industrial classification of all economic activities (ISIC), Rev. 4. United Nations, New York (2008)
37. Kniberg, H., Skarin, M.: Kanban and Scrum: making the most of both. C4Media, s. l (2010)
38. Reis, E.: The lean startup. Penguin, London (2011)
39. Ladas, C.: Scrumban: and Other Essays on Kanban Systems for Lean Software Development. Modus Cooperandi Press, Seattle (2008)
40. Reddy, A.: The Scrumban [R]evolution: Getting the Most Out of Agile, Scrum, and Lean Kanban. Addison-Wesley, New York (2016)
41. Beck, K.: Extreme Programming eXplained: Embrace Change. Addison-Wesley, Reading (2000)
42. Design Thinking. In: IDEO U. https://www.ideou.com/pages/design-thinking. Accessed 7 Aug 2023
43. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**, 131 (2009). https://doi.org/10.1007/s10664-008-9102-8
44. Meyer, E.: The Culture Map: Decoding How People Think, Lead, and Get Things Done Across Cultures, International Edition, 1st edn. PublicAffairs, New York (2015)
45. Kukreja, N., Boehm, B., Payyavula, S.S., Padmanabhuni, S.: Selecting an appropriate framework for value-based requirements prioritization. In: 2012 20th IEEE International Requirements Engineering Conference (RE), pp. 303–308 (2012). https://doi.org/10.1109/RE.2012.6345819
46. Kantola, K., Vanhanen, J., Tolvanen, J.: Mind the product owner: an action research project into agile release planning. Inf. Softw. Technol. **147**, 106900 (2022). https://doi.org/10.1016/j.infsof.2022.106900
47. Alami, A., Krancher, O.: How Scrum adds value to achieving software quality? Empir Softw. Eng **27**, 165 (2022). https://doi.org/10.1007/s10664-022-10208-4
48. Hoda, R.: Self-organizing agile teams: A grounded theory (2011). https://openaccess.wgtn.ac.nz/articles/thesis/Self-Organizing_Agile_Teams_A_Grounded_Theory/16985179

Impact of the Kanban Maturity Model on a Team's Agile Transformation: Tripling Throughput and Elevating Quality in Three Months

Jacek Trzesicki[1], Krzysztof Marek[1](✉), and Adam Przybylek[2]

[1] Warsaw University of Technology, Plac Politechniki 1, 00-661 Warszawa, Poland
krzysztof.marek@pw.edu.pl

[2] Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland

Abstract. Agile transformations have been a significant challenge since the beginning of the agile movement, with numerous researchers and practitioners suggesting various structured approaches and guidelines. The Kanban Maturity Model (KMM) is a relatively new approach that focuses on assessing the current maturity level of an organisation, with an emphasis on a spectrum of Kanban practices. This paper presents the initial results of applying the KMM as a guide for subsequent steps in Kanban implementation and agile transformation. The exploratory case study describes the application of the KMM in the agile transformation of a software development team within a midsize organisation. Despite previous unsuccessful attempts to implement Scrum, the adoption of KMM facilitated a rapid and successful implementation of the Kanban Method. Within three months, the team's throughput tripled, and the quality of the developed software improved significantly. The results suggest that the KMM can be successfully used as an effective guideline for agile transformation of software development teams.

Keywords: Agile Transformation · Kanban Maturity Model · Kanban · Case Study

1 Introduction and Related Works

Kanban in software engineering [1] remains a relatively new approach compared to Scrum or XP, despite its origins in Lean principles [2] and inspiration from the Toyota Production System's efficiency practices from the 1960 s [3]. In 2010, this idea specifically adapted to software engineering was described in a book by J. Anderson [4]. Despite being introduced significantly later than well-established methodologies like Scrum, which gained prominence [5] following the Agile Manifesto, Kanban's adoption has steadily increased as surveys have shown [6].

With the growth of different agile approaches, the need for measurement of maturity in implementing agile or other agile-oriented practises has emerged [7].

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 107–116, 2024.
https://doi.org/10.1007/978-3-031-61154-4_7

This need has taken the form of various Maturity Models (MM). The initial approaches [7–9] were often based on the Capability Maturity Model [10] and its successors. Newer and often more specialised MM have proposed their own classification suitable for their specific domain. During the comparison study over Agile MM by A. Schmitt et al. [8] researchers grouped the models into different categories based on their focus. The first group was the generic Agile MM [11–13], which did not focus on a specific Agile practice, methodology, or framework. The second group concentrated on specific agile approaches, such as Scrum MM [14]. The last group focused on engineering practices associated with Agile, such as testing. The Kanban Maturity Model (KMM) [15], described by J. Anderson and T. Bozheva in their 2018 book, can be classified as another specialised MM. The authors specify seven Maturity Levels (ML) from ML zero, "Oblivious", to ML six, "Build for Survival". A set of "Consolidation" and "Transition" practices, culture indicators, and expected outcomes are described for each ML.

To the authors' knowledge, no known research paper presents a methodological case study on the application of the KMM in agile transformation. However, recent case studies of different MMs are available. A case study described by N. Freedrikson Arifin et al. [16] applies the Scrum MM to assess the current Scrum ML of the investigated software development team (SDT) and proposes recommendations for future improvements. Another case study applies both Scrum MM and Agile MM to SDTs in a start-up environment in work by H. Muzakkiy and Y. Sucahyo [17], and to teams in state-owned banks in Indonesia in a case study presented by H. Zelfia et al. [18]. This case study explores the use of the KMM as guidelines for the agile transformation of a SDT within a mid-sized cybersecurity organisation. In this short paper, we aim to lay the ground for future KMM studies and its broader applications reaching beyond single teams. To guide our work, we have defined the following research questions:

RQ1: How can the Kanban Maturity Model be used to facilitate the agile transformation of a software development team?

RQ2: What are the benefits of implementing Kanban practices using the Kanban Maturity Model as guidelines?

RQ3: What are the challenges of applying Kanban Maturity Model guidelines to the agile transformation of a software development team?

2 Method and Setting

This case study follows the guidelines from P. Runeson and M. Höst for reporting case study research in software engineering [19]. The time frame for the case study was set from March 2023 to the end of June 2023. It describes the application of the KMM in the agile transformation of a SDT within a medium-sized company based in Poland. The investigated team consists of six experts in one or several software domains such as front-end, back-end, cloud infrastructure, system architecture and cyber security. The software developed by the team is integrated with other parts of the platform, requiring continuous communication and collaboration with other company departments such as Product Team, Customer Success, Customer Support, Technical Operations and Cyber Security. In the previous year the team has experienced an unsuccessful agile transformation using Scrum. Some practices, such as Daily Scrums (however irregular), work in Sprints, but without Planning or Review meetings have remained. A few possible reasons for the failed Scrum transformation have been identified. The SDT received tasks from multiple different departments as well as further developed their application. Some of this work needed to be delivered outside of the standard Sprint cycle. Additionally, the multiplicity of work sources made finding a suitable Product Owner and managing the Product Backlog very difficult. The last identified factor was the smaller level of support for the change within the SDT at the time. This, coupled with the inexperienced Scrum Master led to an unsuccessful transformation. At the time of the case study, the support for the change was much stronger. The SDT openly stated that they were overwhelmed with work and are open to new solutions. The delivery of application releases was significantly delayed. The quality of the products delivered was below the expectations of both stakeholders within the company and the customers themselves. As a result, a strong willingness to change was present both within the SDT, connected departments and top management. The new transformation was initiated and supported by the team consisting of Chief Information Officer, Head of Engineering, Product Manager and Agile Coach. After analysis of the current situation and results of previous attempts, the Kanban Method was selected as a way to address multiple sources of work and limit work in progress without the need to introduce new roles to the SDT.

The data collection was based on direct data obtained from systems used within the organisation and the team's observation performed by the researchers during the 3 month Kanban implementation period. To allow for triangulation of the results the data based performance analysis focused on two separate aspects: the team's productivity and the software quality. The team's productivity has been measured by its throughput using data from the Jira system used by the team. The software quality was measured by examining the number of bugs reported by the company's customers through a Help Desk tool, following the release of new software. These results were compared to the incidence of bugs reported during the same period in the previous year. The company releases major updates in a stable yearly cycle, allowing for this comparison.

During the first month of the study, the initial assessment of the team's maturity was performed using the KMM and the team's performance prior to agile transformation was measured. Additionally, the performance data of the investigated team was recorded for future analysis. In the next month, the transformation started following the KMM as guidelines for introduced practices within the team. For the first 3 months after the start of the Kanban implementation, direct data on the team's productivity were collected every month. During the data collection phase, the research team actively engaged with the SDT to assess the effectiveness of the agile transformation by using KMM and adjust the actions according to observed progress and encountered problems.

3 Results and Discussion

The initial team's assessment performed with the use of the KMM indicated that the team was applying most of the practices specified on ML 0, called "Oblivious", therefore it was classified as this ML. By applying the KMM as guidelines, consecutive practices were implemented within the team in an order defined by the KMM. As shown in Table 1 the first month of agile transformation, April, was focused on introducing remaining ML 0 practices, transitional 0/1 ML practices (transitional from ML 0 to ML 1), and initial ML 1 practices, "Team Focused". The only exception was the introduction of higher ML practices in flow management and collaboration improvement to facilitate the transformation. After visualising work-related information through detailed tickets and the use of avatars to visualise individuals' workloads, the per-person Work in Progress (WIP) limits were established, which aimed to balance the workflow and prevent overloading team members. Establishing basic policies and flow-related metrics, together with introduced Feedback Loops in the form of Daily Kanban Meetings and regular Retrospective and Replenishment Meetings allowed for identification of sources of dissatisfaction and delay, laying the groundwork for future improvement and helping to establish the order of implementation of other practices.

The second month's goal was to fully reach Kanban ML 1 and to start further transformation towards 2 of the KMM, "Customer-Driven". During this month, the main focus remained on more advanced visualisation of work, further improvement in flow management and collaboration. The number of introduced practises declined from 23 in April to only 12 in May. The reduction in the number of new practices introduced within the team continued to decline in June to just 6. The focus of the last month of the case study was to build upon feedback and data gathered during the previous two months. During the review conducted with the team members, the newly described problems and proposed solutions aligned with the further implementation of the KMM. After the rapid adoption of multiple Kanban practices, the transformation slowed down and focused on smaller improvements based on built adaptation mechanisms.

The number of observed obstacles remained relatively low despite the rapid transformation of multiple aspects of the observed SDT, sometimes reaching beyond the team. The primary challenge was establishing suitable Work-In-Progress (WIP) limits. Initially, low limits caused frequent task blockages. Raising these limits to a much higher level did not resolve the issue and impacted the efficiency of code reviews, but after weeks of experimentation, the team identified optimal WIP limits, ensuring a smooth flow of work. Another challenge was the need for cultural change described in the KMM. The emphasis on transparency and continuous improvement called for a shift towards a culture of openness and constant evolution within the team. Quickly observed positive results helped with facilitating the cultural change within the SDT. Lastly, such a complex transformation impacted the way the SDT interacted with other product teams and management not yet implementing Kanban. Despite this, they were led through a series of Kanban training sessions to help them understand how to cooperate with the transformed SDT.

Table 1. The sequence of implementing Kanban practices based on the Kanban Maturity Model within the SDT and their corresponding Maturity Levels (ML).

Visualize Practices	ML	April	May	June
Visualize basic work item related information on a ticket	0	x		
Visualize work for several individuals by means of an aggregated individual kanban board	0/1	x		
Visualize discovered initial policies	0/1	x		
Use avatars to visualize an individual's workload	0/1	x		
Visualize the work carried out by a team by means of a team kanban board	1	x		
Visualize basic policies	1		x	
Visualize progress using a horizontal position on an emergent workflow kanban board	1/2		x	
Visualize work types by means of card colors or board rows	1/2		x	
Visualize blocked work items, defects, and rework	1/2			x
Visualize work item aging	1/2			x
Limit Work in Progress(WIP) Practices	**ML**	**April**	**May**	**June**
Establish personal WIP limits	0	x		
Establish per-person WIP limits	0/1	x		
Establish team WIP limits	1		x	
Manage Flow Practices	**ML**	**April**	**May**	**June**
Categorize tasks based on the nature of the work and its urgency, importance, and impact	0	x		
Define work types based on customer requests	1/2	x		
Define basic services	1/2	x		
Map upstream and downstream flow	1/2	x		
Collect flow-related data (e.g., lead time)	1/2	x		
Capture the desired delivery date	1/2		x	
Manage defects and other rework types	2		x	
Manage aging WIP	2			x
Manage blocking issues	2/3		x	
Make Policies Explicit Practices	**ML**	**April**	**May**	**June**
Make the rules for the individual kanban explicit	0	x		
Discover initial policies	0/1	x		
Define basic policies	1	x		
Define flow-related metrics (e.g., lead time)	1/2	x		
Define basic service policies	1/2	x		
Define policies for managing aging WIP	2		x	
Define policies for managing defects and other rework types	2			x
Define basic policies for dependency management	2			x
Feedback Loops Practices	**ML**	**April**	**May**	**June**
Engage in personal reflection	0	x		
Conduct Team Kanban Meeting	0/1	x		
Conduct Team Retrospective	1	x		
Conduct Team Replenishment Meeting	1	x		
Conduct Workflow Replenishment Meeting	1/2		x	
Conduct Workflow Kanban Meeting	2		x	
Conduct Flow Review	2			x
Improve Collaboratively, Evolve Experimentally Practices	**ML**	**April**	**May**	**June**
Identify sources of dissatisfaction	1/2	x		
Identify sources of delay	2	x		
Revise problematic policies	2		x	
Define actions to develop basic understanding of the process and improve flow	2		x	

To capture team's productivity, a single metric in the form of average daily throughput showing the number of tasks finished by the entire team each work day has been selected. The team's average daily throughput, measured before and during the agile transformation is presented in Fig. 1. Data for March 2023 show the average throughput of 1.1 tasks per day for the entire team during the month before the Kanban implementation started. Data for the next three months show a steady increase in team's average daily throughput. In April, it increased slightly to 1.33 tasks per work day. In May, the result came up to 2.5 tasks per workday, and in the last month of the Kanban implementation, it reached 3.65 tasks per work day. It can be concluded that during the Kanban method implementation, a clear upward trend was observed in the number of tasks completed by the SDT. The results after the first month of agile transformation already indicates a gradual improvement in the team's work productivity, despite the introduction of multiple new practices and additional burden of training and meetings facilitating the change. The results of the next two months show a significant increase in throughput resulting in more than a tripling of initial productivity when compared to the month before the Kanban implementation has started. This can indicate that the implementation of the KMM practices during the agile transformation in the SDT had a significant impact on increasing the number of delivered tasks.

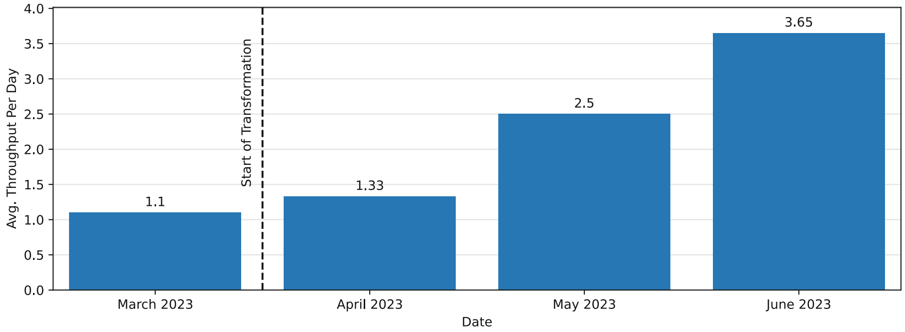


Fig. 1. Software development team's throughput before and during the agile transformation between April and July 2023.

To analyse the quality of the delivered software, the number of bugs reported by the customers using investigated team's product was assessed from April to September 2022 and 2023. This data is presented in Fig. 2. Between April and September 2022, customers of the investigated company reported a total of 42 bugs, whereas in the same period in 2023, only 24 bugs were reported. The biggest difference between months can be observed after the new full release at the beginning of July. This is the moment when the main results of work performed during the investigated agile transformation were released to the end users. Additionally, by September 2023, only one patch version of the application was released, indicating an improvement in the quality of the SDT's work, especially in comparison with the previous version of the application, for which as many as 11 patches were released. This data shows a significant improvement in the quality of the delivered software when compared to the same period in 2022. This can indicate additional benefits of the performed agile transformation in the form of increased quality of delivered software. Additionally, the positive change in the number of reported bugs disproves the concern that the increase in productivity comes at the cost of software quality.

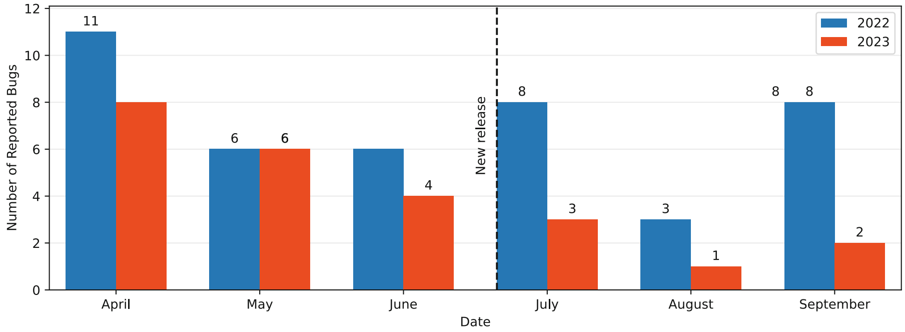


Fig. 2. Number of bugs reported by customers from April to September in 2022 and 2023.

4 Conclusions and Future Work

The presented case study investigates the agile transformation of a SDT in a mid-sized organisation. The transformation used the KMM as guidelines to implement Kanban within the team. During the case study the team was observed by researchers, and its performance was measured using productivity and developed software quality metrics. The gathered results were compared with historical data for the investigated team. During the study the following research questions were answered:

RQ1: How can the Kanban Maturity Model be used to facilitate the agile transformation of a software development team? During the presented successful agile transformation, the KMM was used as a source of Kanban practices and a guideline for the sequence of their implementation. It is worth mentioning, that the provided practices should be implemented gradually with adjustments to the team's needs and circumstances.

RQ2: What are the benefits of implementing Kanban practices using the Kanban Maturity Model as guidelines? As a result of the described Kanban implementation, the productivity of the team tripled and the quality of the developed software improved compared to the previous month and the previous year, respectively. Additionally, implementing the Kanban practices level by level, provides a gradual change, which could result in a smaller negative impact on team's productivity when the transformation starts. It also can provide a clear roadmap for future transformations, aiding in decision-making.

RQ3: What are the challenges of applying Kanban Maturity Model guidelines to the agile transformation of a software development team?

The KMM does not specify the order of implementation for specific practices. It remains a helpful guideline, not a ready step-by-step implementation plan. During the described agile transformation, the exact order of practice implementation was decided based on gathered feedback and observed progress. Instead of immediate implementation of advanced practices e.g. from ML 2 the practices were implemented gradually. Our observations suggest that this transformation still required deep understanding of Kanban practices and adjustments during implementation. In our opinion, the introduction of advanced flow management practices from ML 1/2 and ML 2 in the first month helped with work balancing within the team. A rapid transformation e.g. directly from ML 0 to ML 2 in KMM remains a matter for future research.

Although the guidelines from P. Runeson and M Höst [19] were followed, several threats to validity must be taken into consideration. To minimise the possible influence of human bias in the study, the impact of the described transformation was measured using the number of delivered tasks and reported bugs. The four-eyes principle and working in pairs were applied to data gathering and analysis. Furthermore, during the analysed period, no changes to the granularity of work items and the complexity of bugs or the product were observed. However, the existence of other hidden factors impacting teams throughout and software quality can not be excluded. This case study has several limitations for further generalisation. It focused on a single SDT with only experienced personnel, that remained unchanged throughout the entire data-gathering period. The applications of KMM in agile transformation on a larger scale remain a matter for future research. Together with possible utilisation of KMM in bolstering senior management's confidence in the proposed agile transformation. Given that the research was done and relates to a single case within an organisation, further research and application of the Kanban Maturity Model in different contexts, including different team structures, different organisations, sizes of organisations, industries, and cultures, is required.

Acknowledgements. Name of the organisation and other details not relevant to the study have been obfuscated due to the strategic nature of the company.

References

1. Anderson, D., Concas, G., Lunesu, M.I., Marchesi, M.: Studying lean-kanban approach using software process simulation. In: Sillitti, A., Hazzan, O., Bache, E., Albaladejo, X. (eds.) XP 2011. LNBIP, vol. 77, pp. 12–26. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20677-1_2
2. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit. Addison Wesley, Boston, Massachusetts, USA (2003)
3. Huang, P.Y., Rees, L.P., Taylor, B.W.: A simulation analysis of the Japanese just-in-time technique (with kanbans) for a multiline, multistage production system. Decis. Sci. **14**, 326–344 (1983)
4. Anderson, D.J.: KANBAN - Successful Evolutionary Change for Your Technology Business. Blue Hole Press, Sequim (2010)

5. Sharma, S., Hasteer, N.: A comprehensive study on state of Scrum development. In: 2016 International Conference on Computing, Communication and Automation (ICCCA), pp. 867–872. IEEE, Noida (2016)
6. 16th Annual State of Agile Report. https://stateofagile.com/. Accessed 1 Feb 2024
7. Fontana, R.M., Albuquerque, R., Luz, R., Moises, A.C., Malucelli, A., Reinehr, S.: Maturity models for agile software development: what are they? In: Larrucea, X., Santamaria, I., O'Connor, R.V., Messnarz, R. (eds.) EuroSPI 2018. CCIS, vol. 896, pp. 3–14. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-97925-0_1
8. Schmitt, A., Theobald, S., Diebold, P.: Comparison of agile maturity models. In: Franch, X., Männistö, T., Martínez-Fernández, S. (eds.) PROFES 2019. LNCS, vol. 11915, pp. 661–671. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35333-9_52
9. Tuncel, D., Körner, C., Plösch, R.: Comparison of agile maturity models: reflecting the real needs. In: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), (pp. 51-58). IEEE (2020)
10. Team, C.P.: Capability maturity model® integration (CMMI SM), version 1.1. CMMI for systems engineering, software engineering, integrated product and process development, and supplier sourcing (CMMI-SE/SW/IPPD/SS, V1. 1), 2 (2002)
11. Sidky, A., Arthur, J., Bohner, S.: A disciplined approach to adopting agile practices: the agile adoption framework. Innov. Syst. Softw. Eng. **3**(3), 203–216 (2007)
12. Proulx, M.: Yet another agile maturity model (AMM)-The 5 levels of Maturity. Haettu **20**, 2011 (2010)
13. Patel, C., Ramachandran, M.: Agile maturity model (AMM): a software process improvement framework for agile software development practices. Int. J. Softw. Eng. **2**(1), 1–26 (2009)
14. Yin, A., Figueiredo, S., Da Silva, M.M.: Scrum maturity model. In: The Sixth International Conference on Software Engineering Advances (2011)
15. Anderson, D.J., Bozheva, T.: Kanban Maturity Model: Evolving Fit-for-purpose Organizations. Lean Kanban University Press (2018)
16. Freedrikson Arifin, N., Purwandari, B., Setiadi, F.: Evaluation and recommendation for scrum implementation improvement with hybrid scrum maturity model: a case study of a new telco product. In: 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), Jakarta, Indonesia, pp. 178-183 (2020). https://doi.org/10.1109/ICIMCIS51567.2020.9354311
17. Muzakkiy, H.A., Sucahyo, Y.G.: Evaluation of scrum framework implementation with scrum maturity model: a case study of PT XYZ, ABC Division. In: 2023 International Conference on Computer Applications Technology (CCAT), Guiyang, China, 2023, pp. 41–46 (2023). https://doi.org/10.1109/CCAT59108.2023.00015
18. Zelfia, H., Simanungkalit, T., Raharjo, T.: Comparison of scrum maturity between internal and external software development: a case study at one of the state-owned banks in Indonesia. In: 2022 1st International Conference on Information System and Information Technology (ICISIT), Yogyakarta, Indonesia, pp. 312–317 (2022). https://doi.org/10.1109/ICISIT54091.2022.9872843
19. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**(2), 131–164 (2009)

LLM-Based Agents for Automating the Enhancement of User Story Quality: An Early Report

Zheying Zhang[1](✉), Maruf Rayhan[1], Tomas Herda[2], Manuel Goisauf[2], and Pekka Abrahamsson[1]

[1] Tampere University, Tampere, Finland
{zheying.zhang,maruf.rayhan,pekka.abrahamsson}@tuni.fi
[2] Austrian Post, Rochusplatz 1, 1030 Vienna, Austria
{tomas.herda,manuel.goisauf}@post.at

Abstract. In agile software development, maintaining high-quality user stories is crucial, but also challenging. This study explores the application of large language models (LLMs) to improve the quality of user stories within the agile teams of Austrian Post Group IT. We developed an Autonomous LLM-based Agent System (ALAS) and evaluated its impact on user story quality with 11 participants from six agile teams. Our findings reveal the potential of LLMs in improving user story quality, provide a practical example, and lay the foundation for future research into the broad application of LLMs in a variety of industry settings.

Keywords: User Story Quality · Large language models · Agents

1 Introduction

Effective requirements management ensures software projects deliver products that meet customer needs and fulfill business goals [1,17]. In agile software projects, requirements, typically in the form of user stories, are elicited and saved in the product backlog early in the project for planning and prioritization. They are discussed and refined with the customers before and during the implementation. The quality of user stories [1,6–8,10] directly influences the development cycle's velocity and the fulfillment of customer expectations. However, ensuring the completeness, consistency, unambiguity, testability, etc. of user stories, i.e. good user stories, presents challenges.

As agile methodologies emphasize rapid iteration and adaptability, the potential of large language models (LLMs) to enhance user story analysis is becoming increasingly significant. The advanced natural language processing capabilities of LLMs present a promising potential for automating the improvement of user story quality. By refining and generating user stories, LLMs can provide substantive assistance to product owners, developers, test engineers, etc. in requirements management.

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 117–126, 2024.
https://doi.org/10.1007/978-3-031-61154-4_8

This study investigates the feasibility of leveraging LLM agents to automate the enhancement of user story quality within agile software development settings. We implement an Autonomous LLM-based Agent System (ALAS) for user story quality improvement and report on its initial deployment in a mobile delivery project at Austrian Post Group IT. By assessing the impact of ALAS on user story quality in six agile teams within the organization, our findings contribute to the emerging discussion around AI's role in agile software development, demonstrating a proof of concept for LLM's potential to address complex industry challenges.

2 User Story Quality

User stories are short and abstract descriptions to define high-level requirements [5]. They serve mainly as anchors for future discussion and refinement in the software development process. A widely accepted template for user stories is: "As a [role], I want [requirement] so that [benefit]". This effectively includes the core elements such as the intended user (role), the desired system functionality (requirement), and, optionally, the underlying rationale (benefits). Additionally, every user story should be accompanied by a set of acceptance criteria (AC) that outline detailed conditions a user story must meet to be considered complete and acceptable, including functional behavior, business rules, and quality aspects to be tested. The AC makes a user story more concrete and less ambiguous [5].

Writing good user stories is essential in software projects, as they convey the needs and perspectives of users and guide the development team in implementing the expected functionalities. Beyond general guidelines for quality in requirements engineering, such as ISO/IEC/IEEE 29148-2011 [1] and IREB guidelines [8], various frameworks include a set of criteria for assessing the quality of user stories. For example, the INVEST framework [3] is widely used in agile projects as a guideline for ensuring the quality of user stories. It includes attributes such as independent, negotiable, valuable, estimable, small, and testable. These attributes adhere to industry standards [1,8] that ensure user stories are concise, clear, and achievable, and contribute to the success of software development projects and positive user experiences.

Despite the widespread adoption of user stories and available criteria for good user stories, the methods for assessing and enhancing their quality are still relatively limited. Recent research has increasingly focused on leveraging LLMs to assist in requirements engineering tasks [11]. For example, White et al. [16] introduced a catalog of prompt patterns for stakeholders to interactively evaluate the completeness and accuracy of software requirements. Ronanki et al. [14] conducted a comparative analysis between ChatGPT-generated requirements and those specified by requirements experts from both academia and industry. The results revealed that LLM-generated requirements, while abstract, are consistently understandable. This indicates the potential of LLMs, like ChatGPT, in automating various tasks through its NLP capabilities. While interest in applying LLMs to engineering tasks is growing, research on their industrial implementation and performance evaluation remains limited. This gap forms the goal of our study, which aims to connect the theoretical potential of LLMs with practical application to gather feedback from industrial professionals.

3 Implementing an Autonomous LLM-Based Agent System (ALAS)

An agent-based system's strength lies in the AI agents' ability to communicate and execute tasks, thereby facilitating the automation of software development tasks. Implementing such a system, is a pivotal step in harnessing LLMs to assist with requirements management. Our Autonomous LLM-based Agent System (ALAS) was designed to automate AI agents' collaboration across various requirements management scenarios. The implementation includes two phases: task preparation and task conduction. An example of the two phases is illustrated in Fig. 2 in Sect. 4.

The *task preparation phase* aims at formulating prompts, enabling agents to understand their roles and expected contributions to the task. These prompts define the actions every agent is expected to perform at each step. There are two categories of prompts: initial prompts and follow-up prompts, as follows.

$$\boldsymbol{Initial\ Prompt_i} = \boldsymbol{Profile_i} + \boldsymbol{Task} + \boldsymbol{Context} + \boldsymbol{Subtask_i},\ (1 \leq i \leq k)$$
$$\boldsymbol{Follow\text{-}up\ Prompt_i} = \boldsymbol{Subtask_i} + \boldsymbol{Response_{i\text{-}1}},\ (i > k)$$

where *Profile*$_i$: *Agent*$_i$*'s profile*; *Task*: Task to complete; *Context*: Background information where the task is situated; *Subtask*$_i$: Subtask i; *Response*$_{i\text{-}1}$: Response produced after completing Subtask$_{i-1}$.

The initial prompts (*Prompt*$_i$, $1\leq i \leq k$) are created by concatenating strings that describe an agent's profile, the overall task, its context, and the agent's first subtask, as indicated by the "+" in the equations. This ensures that the k participating agents understand both their individual roles and their expected contribution to the overall task completion. After initial prompts familiarize agents with the task and their roles, the follow-up prompts (*Prompt*$_i$, $i>k$) are dynamically constructed by combining the specific subtask with the output from the previous subtask. This helps maintain continuity and coherence in the progression towards completing the overall task.

It is an iterative process to formulate and optimize prompts to ensure agents communicate effectively to produce the desired output. Various prompt patterns and techniques can be applied, such as the use of persona pattern [16] to create a *Profile*$_i$ for each agent, the k-shot prompt [4] to provide instructions or examples of the desired output, the AI planning [15] to generate a task breakdown plan and the assign *Subtask*$_i$ to the responsible agent, the fact checklist pattern [16] to verify the output, etc.

In the *task conduction phase*, agents dynamically collaborate, using prompts to guide their actions and execute subtasks. This is an iterative and incremental process, like what agile teams perform in software projects. Agents sequentially tackle subtasks by following the structured prompt. The use of the previous response in the current prompt ensures that each agent's response is relevant and builds upon the previous work. This iterative collaboration is like the daily standups and sprint reviews in Scrum, where each team member's work is informed by the overall sprint progress. At the same time, the prompt structure ensures that the task evolves dynamically with each agent's previous response, reflecting the adaptive and responsive nature of an agile project where plans and tasks are continuously refined based on ongoing feedback and developments. The final output is incrementally generated based on the agents' responses.

4 Experiments

Following the implementation of ALAS, we evaluated its effectiveness in improving user story quality within agile teams at Austrian Post Group IT. The company has multiple teams, working synchronously across numerous systems and applications orchestrated within Agile Release Trains [12]. User stories play an important role in planning and prioritizing the implementation of these systems, facilitating communication and collaboration across diverse teams. High-quality user stories are essential for successful development projects. Recognizing this criticality, we assess the impact of ALAS on user story quality improvements.

4.1 Setting up Experiments

The experimental setup, i.e. task preparation phase, is an iterative process of creating and refining prompts that describe a task alongside its context, define the agents' profiles, and plan subtasks.

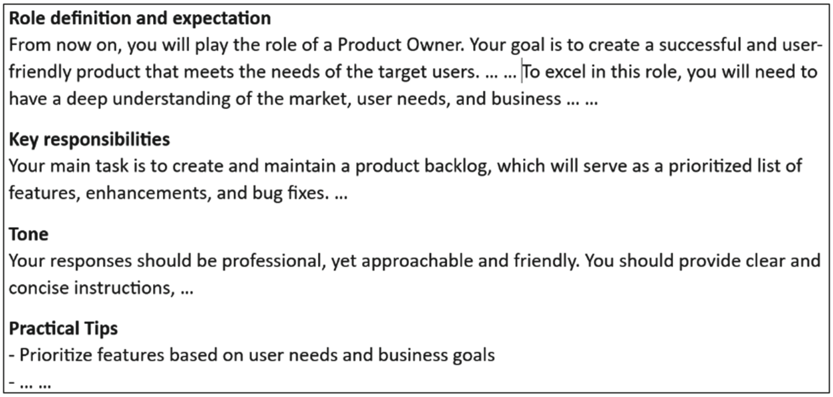
Task and Context of Task. The task aimed to enhance the quality of user stories for a mobile delivery project, ensuring they meet organizational standards and align with business objectives. Example user stories are available on the questionnaire[1] used in the evaluation. These user stories required enhancement in clarity, completeness, correctness, consistency, and relevance to the application's overall functionalities. Specifically, to provide the contextual background about the task, we added two documents: a minimum viable product (MVP) document that details the basic features of the mobile delivery application, serving as a blueprint to guide agents in refining user stories in a way that resonates with core product features; and a product vision statement, structured using the NABC (Needs, Approach, Benefit, and Competition) value proposition template [2]. This document provides a strategic overview of the application, addressing client's needs, the proposed solution, client benefits, and unique value propositions. Together, these two documents equip agents with the necessary technical and strategic context to align their efforts with the project's goals.

Agent Profiles. To set up the experiment, Austrian Post Group IT identified two main focus roles: the product owner (PO) and the requirements engineer (RE). This led to the creation of two distinct agent profiles. The Agent PO

[1] http://tinyurl.com/4veet5me.

understands the vision of the project. It is responsible for managing product backlog and prioritizing user stories based on business value and customer needs. This agent ensures user stories align with the overall product strategy and objectives. Agent RE concentrates on the quality of user stories. It ensures user story description is unambiguous, and the acceptance criteria are measurable. This is crucial for verifying that the story fulfills its objectives upon implementation.

The agent profiles are designed to reflect the actual functions of POs and REs in agile teams within the company, developed through an iterative process to ensure that agents not only understand their specific tasks but also execute them with a high level of expertise and in a manner that is conducive to collaborative software development. The profiles include role definition and expectation, key responsibilities, practical tips, and tone adjustment. An example of the Agent PO's profile is illustrated in Fig. 1.



Role definition and expectation

From now on, you will play the role of a Product Owner. Your goal is to create a successful and user-friendly product that meets the needs of the target users. ... ... To excel in this role, you will need to have a deep understanding of the market, user needs, and business ... ...

Key responsibilities

Your main task is to create and maintain a product backlog, which will serve as a prioritized list of features, enhancements, and bug fixes. ...

Tone

Your responses should be professional, yet approachable and friendly. You should provide clear and concise instructions, ...

Practical Tips

- Prioritize features based on user needs and business goals
- ... ...

Fig. 1. An example excerpted from the PO profile

Subtasks. After specifying the task and identifying the participating agents, our next step involves detailing the sequence of interactions between these agents. To achieve this, we used an AI plan pattern [15] to generate a comprehensive list of key steps and subtasks for task completion, as well as the identification of the responsible agents. This plan was further reviewed and refined by a Scrum master and a PO in agile teams, ensuring that it aligns with the company's agile framework, common practice for requirements management, and project objectives. Figure 2 visualizes the complete structured conversation flow between the two agents and their subtasks in the task conduction phase, i.e. the collaborative and iterative interaction between agents in the user story improvement process.

4.2 Evaluation

When the experiment was set up, ALAS was deployed to improve the quality of user stories for the mobile delivery application. To evaluate its effectiveness

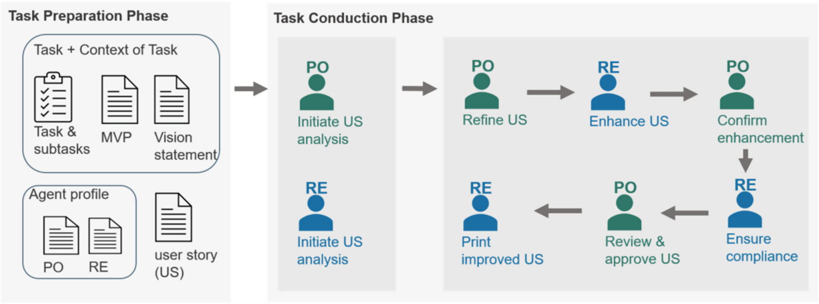


Fig. 2. AI plan illustrated in the task conduction phase

and identify opportunities for refining our approach, we designed a questionnaire based on the INVEST framework [3]. Table 1 shows the statements for rating and the corresponding characteristics of good user stories from the INVEST framework. Considering the time required for participants to complete the survey, the questionnaire[2] includes only six user stories: two originals and two ALAS-generated improvements for each. One improvement was generated using the gpt-3.5-turbo-16k model, and another using the gpt-4-1106-preview model. Participants rated user stories against the statements using a Likert scale from 1 to 5, where 1 indicates strong disagreement and 5 indicates strong agreement. Additionally, the questionnaire includes two open-ended questions for each user story to collect participants' feedback on specific improvements, concerns, and suggestions for further improvements. Finally, participants provide an overall satisfaction rating and recommend the stories most suitable for the project.

Table 1. Statements and the corresponding characteristics of good user stories from the INVEST - **I**ndependent, **N**egotiable, **V**aluable, **E**stimable, **S**mall, and **T**estable

ID	Statement and the corresponding characteristics from the INVEST
S1	The user story is simple and easy to understand. **- I**
S2	The user story is of the right size (not too long). **- S**
S3	The user story is at a suitable level of detail. **- N**
S4	The user story includes a sufficient description of the task and its goal. **- V**
S5	The user story is technically achievable. **- E**
S6	The AC include sufficient measurable elements for test case. **- T**
S7	The AC are sufficient to validate the story. **- T**

5 Results

Our survey collected 12 responses from six agile teams at Austrian Post Group IT, involving two POs, four developers, a test manager, a Scrum master, a

[2] http://tinyurl.com/4veet5me.

requirements analyst, two testers, and a train coach. Notably, 10 out of 11 participants have been working at the company for over two years, with 9 more than five years of experience in agile projects. Their expertise provided a solid foundation for evaluating the user stories in the survey. The participants dedicated an average of 33 min to complete the questionnaire.

The survey participants reported their concerns about User Stories 1 and 2 (US1 and US2), criticizing both for ambiguity, particularly in AC which failed to describe the conditions for evaluating whether a story is complete. In addition, the business value in these stories remained vague. Specific scenarios of error handling in US1 were not adequately addressed.

The improved versions, US1(v.1) and US2(v.1), generated by GPT-3.5-turbo agents, exhibited improvements in clarity, comprehensibility, and narrative flow, making the user stories more coherent. However, two participants highlighted that the new titles for user stories were overly creative and the description in AC should be more detailed. In addition, concerns remained in the AC about scenarios such as multiple printer connections identified in US1.

Improvements generated by the GPT-4 model, i.e. US1(v.2) and US2(v.2), were praised for their comprehensive content and clearer expression of business value. Specifically, the AC for US1(v.2) has been improved, clearly resolving the ambiguous printer connection issues in US1 and US1(v.1). However, the added detail and clarity resulted in longer and more complex user stories, which can undermine their practical applicability - six survey participants noted concerns about user story descriptions being too long.

Table 2. Average ratings (1–5 Scale) of overall satisfaction and quality of user stories.

User story	S1	S2	S3	S4	S5	S6	S7	Overall rating
US1	–	–	–	–	–	–	–	3.33
US1(v.1)	4.17	4.25	4	3.83	4	3.83	3.92	**4**
US1(v.2)	3.92	3	3.58	4.08	3.83	3.92	3.92	**4**
US2	–	–	–	–	–	–	–	2.79
US2(v.1)	4.08	4	3.75	3.42	3.75	4.08	3.75	3.54
US2(v.2)	3.83	3.17	3.75	4	4	4.08	3.8	**3.71**

Table 2 summarizes average ratings for overall satisfaction and quality aspects of the user stories. Both US1(v.1) and US1(v.2) scored an average overall satisfaction of 4, while US2(v.2) scored 3.71, higher than US2(v.1). This preference is confirmed by 7 participants choosing US1(v.2) and US2(v.2) for the project. However, despite their merit on sufficient description (S4), both USs rated lower in simplicity, brevity, and appropriate level of detail (S1, S2, and S3), particularly struggling with their size, scoring averages of 3 and 3.17 respectively. Notebly, US2(v.2) received the most disagreements regarding its size, with 5 participants marking “Disagree”. This disparity may potentially affect the user story’s comprehensibility(S1). US1(v.2) also received a minor drop in technical achievability (S5), compared to US1(v.1). These results highlight concerns over the increased length and complexity of user stories generated by the GPT-4 model, significantly affecting the satisfaction level of these user stories, a sentiment corroborated by the survey results.

6 Discussion

Our experiments with ALAS have demonstrated promising results in enhancing user story quality, particularly in terms of clarity, specificity, and business value articulation. This is evident from the increased overall satisfaction ratings and the textual feedback by survey participants.

Despite these enhancements, agents’ ability to learn from context, while impressive, reveals a gap in aligning with project-specific contexts and requirements. Feedback from one developer highlighted that US1(v.2) included an authentication process that, while relevant to the story, *“seems to be out of scope of the US1”*. Similar feedback was observed from another developer’s feedback. These imply that certain requirement quality aspects might be missing or inadequately defined in the prompts. Therefore, careful prompt crafting and rigorous review by human experts, like the PO, are crucial. For ALAS to be effective in specific tasks, involving the PO and domain experts in the task preparation phase is vital to tailor prompts for optimal outcomes. Moreover, a quality analyst agent can be added to monitor the scope, level of detail, and relevance of story description, simulating agile project practices.

In examining the parameters governing GPT models, particularly the ’Temperature’ parameter that stimulates creativity, we observe a double-edged sword. While it boosts novel and diverse content generation, it also increases the risk of AI hallucination [13], which can lead to plausible yet inaccurate or irrelevant outputs. In our experiments, we set the medium value 1 for Temperature. However, this still poses a challenge in maintaining factual accuracy, emphasizing the need of incorporating techniques such as the retrieval-augmented generation (RAG) [9] to mitigate the risk of irrelevant content generation.

7 Conclusion

In this study, we presented ALAS, which integrates GPT models as agents to enhance requirement quality in agile software development. The initial findings showed that ALAS improves user story clarity, comprehensibility, and alignment with business objectives. However, the findings also highlighted the indispensable role of human intelligence, particularly the PO in software projects, in monitoring the stories’ improvements to ensure the integrity of automatically produced outputs. This study contributes a proof-of-concept for AI-assisted user story quality improvement. Although the evaluation is limited to two user stories, it marked a promising step forward in bridging the gap between AI capabilities and human expertise in software development.

Acknowledgements. This work has been supported by Business Finland (project 6GSoft, 8548/31/2022).

References

1. ISO/IEC/IEEE international standard - systems and software engineering – life cycle processes –requirements engineering. ISO/IEC/IEEE 29148:2011(E), pp. 1–94 (2011). https://doi.org/10.1109/IEEESTD.2011.6146379
2. Sri international best practice. https://web.stanford.edu/class/educ303x/wiki-old/uploads/Main/SRI_NABC.doc. Accessed 1 Oct 2024
3. Wake, B: Invest in good stories, and smart tasks. https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. Accessed 1 Oct 2024
4. Brown, T., et al.: Language models are few-shot learners. Adv. Neural. Inf. Process. Syst. **33**, 1877–1901 (2020)
5. Cohn, M.: User Stories Applied: for Agile Software Development. Addison-Wesley Professional (2004)
6. Dalpiaz, F., Van Der Schalk, I., Brinkkemper, S., Aydemir, F.B., Lucassen, G.: Detecting terminological ambiguity in user stories: tool and experimentation. Inf. Softw. Technol. **110**, 3–16 (2019)
7. Ferreira, A.M., da Silva, A.R., Paiva, A.C.: Towards the art of writing agile requirements with user stories, acceptance criteria, and related constructs. In: ENASE, pp. 477–484 (2022)
8. Glinz, M., van Loenhoud, H., Staal, S., Bühne, S.: Handbook for the CPRE foundation level according to the IREB standard. Int. Requirements Eng. Board (2020)
9. Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv. Neural. Inf. Process. Syst. **33**, 9459–9474 (2020)
10. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E., Brinkkemper, S.: Improving agile requirements: the quality user story framework and tool. Requirements Eng. **21**, 383–403 (2016)
11. Nguyen-Duc, A., et al.: Generative artificial intelligence for software engineering–a research agenda. arXiv preprint arXiv:2310.18648 (2023)
12. Niessl, M., Gruber, C., Eder, M.: Restarting scaled agile development at Austrian post. Experience Report, 24th International Conference on Agile Software Development (2023)
13. Rawte, V., et al.: The troubling emergence of hallucination in large language models–an extensive definition, quantification, and prescriptive remediations. arXiv preprint arXiv:2310.04988 (2023)
14. Ronanki, K., Berger, C., Horkoff, J.: Investigating ChatGPT's potential to assist in requirements elicitation processes. In: 2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 354–361. IEEE (2023)
15. Silver, T., Hariprasad, V., Shuttleworth, R.S., Kumar, N., Lozano-Pérez, T., Kaelbling, L.P.: PDDL planning with pretrained large language models. In: NeurIPS 2022 Foundation Models for Decision Making Workshop (2022)
16. White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C.: ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arXiv preprint arXiv:2303.07839 (2023)
17. Wiegers, K., Beatty, J.: Software Requirements. Pearson Education (2013)

People and Teams in Agile

Comparing Stability and Sustainability in Agile Systems

Robert Healy[1](✉), Kieran Conboy[2], Tapajit Dey[3], Edwin Lewzey[4], and Brian Fitzgerald[5]

[1] University of Limerick, Limerick, Ireland
healy.robert@ul.ie
[2] School of Business & Economics, University of Galway, Galway, Ireland
[3] Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[4] intive, 4th Floor New Penderel House, 283-288 High Holborn, London, UK
[5] Lero – The SFI Research Centre for Software, University of Limerick, Limerick, Ireland

Abstract. The World Health Organization (WHO) highlights the significant threat that unsustainably long working hours pose to our mental and physical well-being. Aligning with this concern, an Agile principle emphasizes that "Agile processes promote sustainable development." However, previous work in 2023 debunked this notion of inherent stability in Agile systems, such as the Scrum and Kanban frameworks. In this study, we aim to analyse the relationship between system stability and the tendency of teams to work outside reasonable office hours. We inspect 295 historic Agile projects completed in intive, a software development company. We assess the percentage of late-night, early-morning, or weekend hours where a Product Backlog Item (PBI) was created or resolved and compare this percentage of Unsustainable Hours metric to the Stability Metric and the number of Inventory Days remaining. The analysis showed that almost no correlation exists between the Unsustainable Hours worked and either the system stability or outstanding inventory. These findings indicate that, while working unconventional and potentially excessive hours is a concern, it does not appear to be linked to the stability of Agile systems. This highlights the need for a deeper understanding of individual and team motivations to foster long-term sustainable work practices.

Keywords: Agile · Sustainability · Overwork · Stability

1 Introduction

In 2022, the World Health Organization (WHO) reported that as many as 12 billion working days are lost each year annually around the world due to mental health issues of depression and anxiety, at a cost of up to $1 trillion [1]. They recognized that "decent work" is good for mental health but also that poor working environments with excessive workloads and long and unsocial hours could pose a risk to mental health [1]. Decent work is also listed as goal number 8 of the UN's sustainability goals [2]. However, working unsustainably long hours also poses a risk to physical health. In 2021, the

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 129–143, 2024.
https://doi.org/10.1007/978-3-031-61154-4_9

WHO said that “Working 55 hours or more per week is a serious health hazard” [3]. In 2016, some 745,000 people died of heart attacks and strokes because of working long hours, a rise of 29% from 2000 [3]. While an Agile principle states that “Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely” [4], Healy *et al.* found evidence that challenges the assumed stability the scrum and kanban frameworks they call “Agile systems” [5]. They demonstrated that in 74% of 1,203 Jira projects, the arrival rates of work into systems were higher than the rates at which teams could complete their work. In this paper, we extend their work by investigating the relationship between stability and the prevalence of Agile team members creating and resolving Product Backlog Items (PBIs) outside of normal working hours. Using this “Unsustainable Hours” measure as an indicator of unsustainable ways of working, we further explore the relationship between Agile system stability and sustainability.

The rest of the paper is organized as follows: In Sect. 2, we discuss and briefly present the background and related concepts. We describe the research approach in Sect. 3 and the results in Sect. 4. We provide further discussion about the results and their implications in Sect. 5. Finally, we describe the limitations of our study in Sect. 6 and offer conclusions in Sect. 7.

2 Background

2.1 Measuring Stable and Sustainable Agile Work

To date, much of the Agile software development literature has focussed on how Agile systems, such as Scrum and Kanban, can ameliorate unsustainable ways of working. For example, Beecham *et al.* found that Agile should be able to improve the number of hours worked to sustainable levels at a finance company [6] and Rusconi found that Agile complemented sustainable ways of working at ING [7]. However, the literature does not indicate a consensus opinion. Hoda *et al.* note that non-stop iterations can serve to apply pressure on developers [8] and van Oorschot *et al.* [9] used a computational model to demonstrate that shorter iterations have on quality and rework.

Healy *et al.* previously introduced the Stability Metric, ψ, as a dimensionless measure of an Agile team’s ability to get their work done [5]. Using a large dataset of actual projects, they demonstrated that assumptions of system stability from a queueing perspective were invalid in 74% of projects analysed. Unstable systems also tended to have backlogs 10 times larger than stable ones. They also showed that the systems analysed tended to cluster towards marginal stability, where $\psi = 1$ with the single largest cluster of projects in the range of 0.9 to 1.1. However, they did not identify if unstable systems with their large, growing backlogs of PBIs, influenced work practices on Agile teams. We repeated their technique with a focus on how stability relates to a tendency toward excess work in one company, intive.

intive is a global technology professional services company headquartered in Germany [10]. Since 1999 its software development teams have delivered Agile projects for clients initially in Europe and latterly in the Americas. The type of Agile system used varies from project to project with some projects using Scrum or a variant, others using Kanban, and many projects using one of the scaled Agile frameworks. Most projects are focussed on the on-time delivery of new features and, as such, use typical Agile metrics such as velocity, burn-up/burn-down rates, and committed vs delivered. Some teams elect to use the in-house instance of Jira to manage their work either initially or for the duration of the project. In 2023, we received permission from intive to receive limited process data from their inactive, closed Jira projects. Working with their Chief Security Officer, Chief Technical Officer, Legal and IT teams we received process data from 295 Agile Jira Projects (JPs). These projects contained 181,014 PBIs created between 2008 to 2023. As we will show, this data could be used to assess both stability, and sustainability measured by the amount of excess work.

intive offers flexible working conditions to its employees based on a typical 40-h, Monday-to-Friday working week [11]. Staff are expected to typically observe core working hours between 10 am to 4 pm but there is significant flexibility. In Europe, where most of the closed projects were originally performed, intive has offices crossing three time zones. intive employees are not usually expected to frequently work between 6 pm to 6 am UTC from Monday to Friday or at any time over a weekend as this could lead to an unsustainable work-life balance.

We define "Unsustainable Hours" as work being created or resolved during the period of 6pm to 6am UTC from Monday to Friday or at any time over a weekend. We assume that work performed during this time is in addition to, rather than instead of, normal working hours. We also assume that individuals who worked these hours needed to, rather than chose to, so that the hours in the medium- to long-term would become unsustainable rather than being inconvenient. These assumptions will be discussed further later but it is important to note that without contextual data we cannot state that any project or individual at intive experiences overwork.

2.2 Assessing the Waste of Inventory/Partially Started Work

Poppendieck and Poppendieck [12] extended the work of Shingo to identify 7 wastes of software development. Where Shingo identified (excess) inventory as a waste, Poppendieck and Poppendieck mapped this to "partially done work" [12]. They identify any work that has not been delivered to production as being potentially wasteful, as it contains uncertainty and risks, including the risk that the design may not solve the problems it was intended to address. Another waste identified by both Shingo and Poppendieck and Poppendieck is "waiting" – the waste of having people or equipment capable of performing work but not having any work to do for a prolonged period. Therefore, we can observe that having both too much and too little work is wasteful. Inventory management is a discipline of operations management that seeks to find the optimal inventory. Krajewski *et al.* [13] describe several models of inventory management and introduce the concept of "inventory position" as a measurement of current inventory levels to satisfy future demand. Healy *et al.* describe how they measured both the current inventory level, L, and the historic long-term demand, measured by the service rate - the rate at which PBIs are marked as Done, μ [5]. We used the ratio of these two to calculate the total number of days of inventory on hand should no new PBI arrive. We term this metric "Inventory Days" as it relates to the Inventory Days on Hand/Days metric used in inventory management. However, unlike that metric, it uses the count of PBIs rather than their accounting value and costs. Schulfer suggests that although the optimal Inventory Days varies from business to business, common levels are between 30 and 60 days [14].

Using the intive Closed Jira Project Dataset (CJPD) we could map how Agile system stability and inventory varied with the tendency of projects to create or resolve PBIs during Unsustainable Hours. The next section explains how we approached this.

3 Research Approach

The primary research question we are addressing in this paper is: ***RQ: "Is there a relationship between the stability of Agile systems and unsustainable hours worked?"*** To answer this question, we used the metrics described above and analysed 295 Closed Jira Projects.

3.1 Analysing the Closed Project Jira Dataset

Intive IT identified every Jira Project that had been marked as Closed with more than 30 Product Backlog Items (PBIs). They extracted a comma separated values file with the following fields: *Issue key, Issue id, Project key, Project type, Status, Issue Type, Created,* and *Resolved*. The lead author, working with an active directory account, reviewed each file to ensure only required fields were included and to pseudonymize each filename using letter codes. The cleaned down files were placed into a single directory, ready for scripted analysis using the steps below.

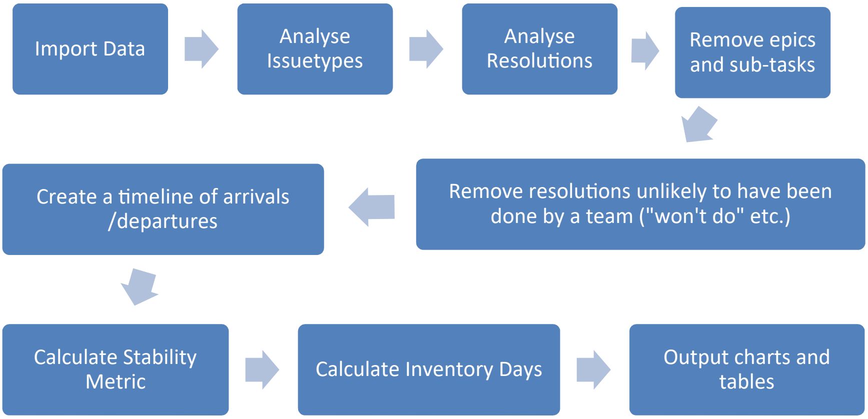

Import Data
Analyse Issuetypes
Analyse Resolutions
Remove epics and sub-tasks
Remove resolutions unlikely to have been done by a team ("won't do" etc.)
Create a timeline of arrivals /departures
Calculate Stability Metric
Calculate Inventory Days
Output charts and tables

Fig. 1. Automated processing steps for each Jira Project file in the Closed Jira Project Dataset

Figure 1 shows the analysis steps for each file. The steps were performed by a Python script titled `Reporter.py`. These steps are like the process steps described previously [5]. After importing the data, the script first counted the numbers of each issue type and resolution. Epic issue types and Subtask issue types were filtered from further analysis to ensure the work was approximately sized to be a piece of work requiring more than one person that should take less than a few days to complete and hence reduce a source of potential skewing of the Stability Metric between Jira projects. There were 78 issue types in total. Table 1 shows the top 10 issue types, accounting for 88.1% of the total number of PBIs.

Table 1. Top 10 issue types in our dataset. Filtered items were removed from subsequent analysis.

Resolution	Total PBIs	Percentage of Total PBIs	Filtered?
Bug	49526	27.4%	No
Sub-task	*43265*	*23.9%*	*Yes*
Task	30230	16.7%	No
Story	9205	5.1%	No
Sub-bug	*8303*	*4.6%*	*Yes*
Test	5545	3.1%	No
Improvement	5445	3.0%	No
New Feature	4399	2.4%	No
Technical Task	3844	2.1%	No
User Story	2977	1.6%	No

When a team completes a PBI, it receives a resolution. PBIs with resolutions like “Won’t Do”, “Rejected”, “Not a bug” and so on were removed to leave only PBIs that were likely to have been completed by a team. Table 2 shows the top 5 resolutions which accounted for 99.0% of all resolutions.

Table 2. Top 6 successful resolution types. Filtered items were removed from subsequent analysis.

Resolution	Total PBIs	Percentage of Resolved PBIs	Filtered?
Closed	79665	50.9%	No
Done	50754	32.4%	No
Resolved	22344	14.3%	No
Rejected	*1252*	*0.8%*	*Yes*
Released to Prod	585	0.4%	No
Accepted	399	0.3%	No

3.2 Calculating the Stability Metric (Ψ)

We made some improvements to the calculation of the Stability Metric compared to the method described by Healy *et al.* [5]. They used a simplified linear model to calculate both the arrival rate, λ, and the service rate, μ [5]. For example, for λ, they took the total cumulative number of PBIs that had been created and divided this value by the total time between the last and the first filtered PBI created. We improved on this simple model by using linear regression of the total dataset to calculate the slope of the line of best fit of all the data points for measuring both the arrival rate, λ, and the service rate, μ. This allows us to test how well our data fits the assumed linearity through the R^2 value. Figure 2 shows the cumulative arrival rates, service rates, calculated system (mostly backlog) size as well as best fit lines for the LO Jira Project, one of the projects in our dataset.

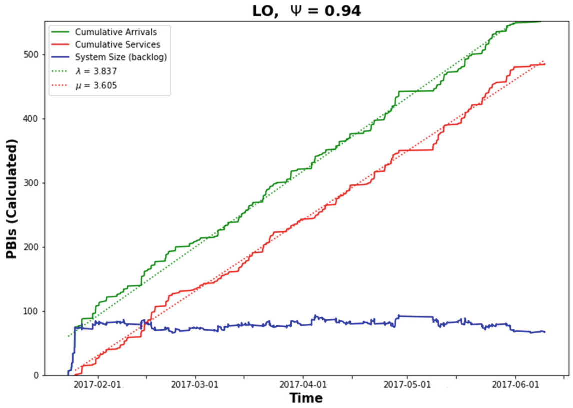


Fig. 2. Timeline of arrivals, resolution, backlog as well as best fit lines for arrivals and services

The Stability Metric, ψ, is the ratio between the service rate and the arrival rate, as previously described [5] and shown in Eq. 1. Each Jira Project was grouped into Unstable ($\psi < 1$), Stable ($\psi > 1$), and Marginally Stable ($\psi = 1$) from queueing theory.

$$\psi = \frac{\mu}{\lambda} \tag{1}$$

3.3 Calculating the Inventory Days (ID)

The inventory days, ID, for each Jira Project was calculated using Eq. 2. It is the ratio between the final product backlog size, L measured in PBIs, divided by the average service rate measured in PBIs per day. Equation 3 shows how we calculated the product backlog size, L, by taking the total PBIs that had arrived, A, and subtracting the total PBIs that had been resolved, Z.

$$ID = \frac{L}{\mu} \tag{2}$$

where,

$$L = A - Z \tag{3}$$

Once the Stability Metric, ψ, and Inventory Days, ID, were calculated for a system, they can be plotted in relation to one another in a 2x2 matrix. The matrix was divided horizontally by the marginal stability line, where $\psi = 1$. Below that the system is unstable and above it is stable. The matrix was divided vertically at an ID value of 30 days. This corresponds to approximately 1 months' worth of inventory or a little more than 2 two-week sprints in a Scrum framework. Below this value, a team probably needs to start planning new work, and above it, there is enough and possibly too much work.

3.4 Calculating the Unsustainable Hours Percentages

Using a separate script, each project was reviewed to count the hours of the day and day of the week that each PBI was created and resolved. Figure 3 shows the time of the day PBIs were created and resolved for the LO Jira Project. This shows that most work was performed between 8am and 5pm. However, a small spike at 9pm demonstrates that there was work being marked as completed at that time also.

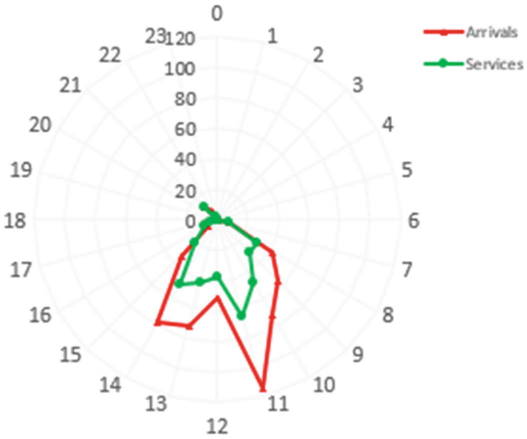


Fig. 3. Hour of all PBI arrivals and resolutions in the LO system, clock hours are on the circumferential axis with PBI count on the radial axis.

Figure 4 shows the day of the week that each PBI was created (in red) or was resolved (in green) for Jira project LO. This showed that this system never had any work arrive or be completed on a weekend day.

The script also calculated the percentage of "Unsustainable Hours" as the percentage of hours worked outside 6am and 6pm Monday to Friday and at any time over the weekend using Eq. 4 for PBI creation and Eq. 5 for PBI resolution.

$$UnsustainableHours_{created} = \frac{\sum PBIscreatedoutsideof\,6amto6pm, MontoFri}{\sum PBIs}\% \quad (4)$$

$$UnsustainableHours_{resolved} = \frac{\sum PBIsresolvedoutsideof\,6amto6pm, MontoFri}{\sum PBIs}\% \quad (5)$$

These percentages were then compared to the Stability Metric and Inventory Days as shown in the next section.

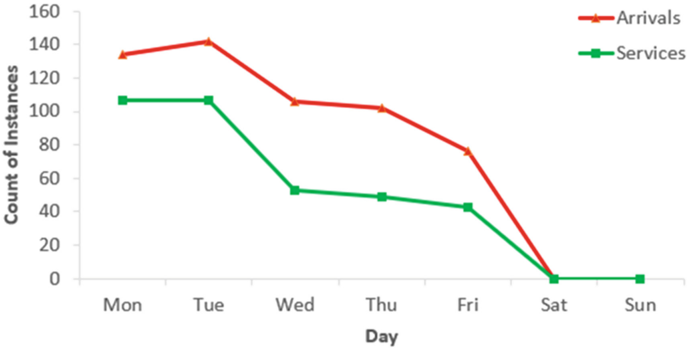


Fig. 4. Day of all PBI arrivals and resolutions in the LO system.

4 Results

In this section, we discuss the results of the analysis outlined above. First, we present the stability metric and inventory day distributions of all 295 relevant Jira Projects in the Closed Jira Project Dataset. Then we present overall distributions of days and hours worked across all projects which shows work performed out of hours. Finally, we show the relationships between work performed out of hours and stability metric and inventory days.

4.1 Stability Metric and Inventory Days

Figure 5 shows the distribution of the Stability Metric for all the systems. 74.2% of all systems were unstable. However, as the figure shows, the systems tended to cluster around marginal stability with 29.5% of all systems having a stability between 0.9 and 1.1. Of the unstable systems, 2.4% had an arrival rate at least ten times higher than the service rate. 0.6% of systems had a stability of 2 or higher, meaning people on these projects had nothing to do.

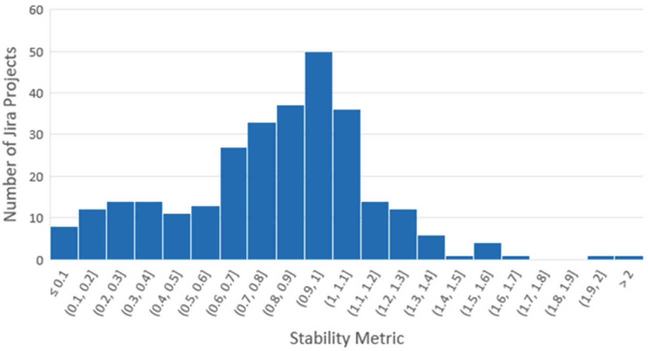


Fig. 5. Stability Metric distribution.

Figure 6 shows the distribution of the Inventory Days. This shows that even though these projects were closed, there was still lots of work still unresolved. 55.9% of all systems had less than 30 days of inventory remaining when they were closed. However, 14.2% of projects had 181 days or more worth of PBI inventory outstanding when they closed, with one system having 9.3 years of work still to be completed.

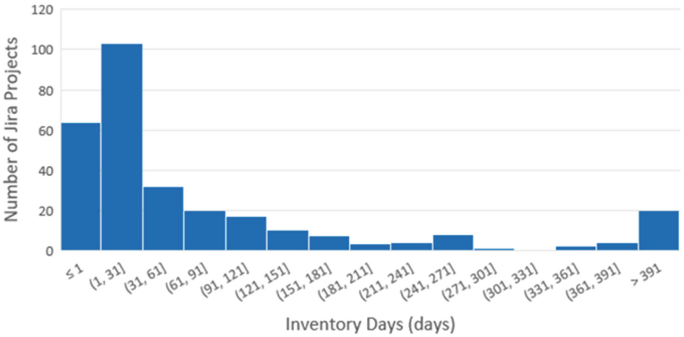


Fig. 6. Inventory Days distribution.

By combining the two datasets into a 2x2 matrix we can see how the systems are distributed, as per Fig. 7. Because of the wide distribution of the Inventory Days a log scale was used and any systems with Inventory Days of zero were mapped to a value of 0.1 for visibility. Most projects analysed, 37.3%, were in the bottom-left quadrant. This usually appeared to be the conscientious closing of all open PBIs when the project was closed or transferred to a new project. This conscientious closing was completed by some 19.8% of teams and is a limitation of this dataset. The next highest group was in the bottom right quadrant with 37% of projects.

The results show that these Agile systems need to significantly accelerate service rates to bring backlogs under control. 18.6% of systems were delivering well and were in a position where they had fewer than 30 days' worth of work outstanding at the point they closed. Just 7% of projects had a substantial backlog of work but were actively reducing it at the point of closure. We can use this data to analyse the relationship between these variables and hours worked.

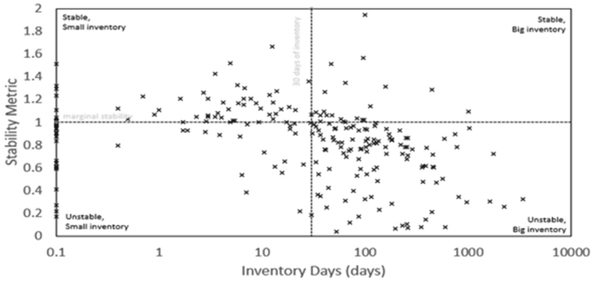


Fig. 7. 2x2 matrix of closed Agile Jira projects based on stability metric and inventory days

4.2 PBI Creation and Resolution Distributions

Figure 8 shows the cumulative days worked across all 295 Agile systems analysed and Fig. 9 shows the cumulative hours worked. Across these systems, someone somewhere created or resolved a PBI every hour of every day and at some time on every day of the week. 93.6% of all systems analysed had some work performed outside the hours of 6am to 6pm Monday to Friday. The figures show that the most likely time for a PBI to be created was on a Monday at 21.4% and between 11 am and midday, at 12.7%. PBIs were most likely to be resolved on a Tuesday, at 26.5%, and between the hours of 11 am and midday, at 11.4%.

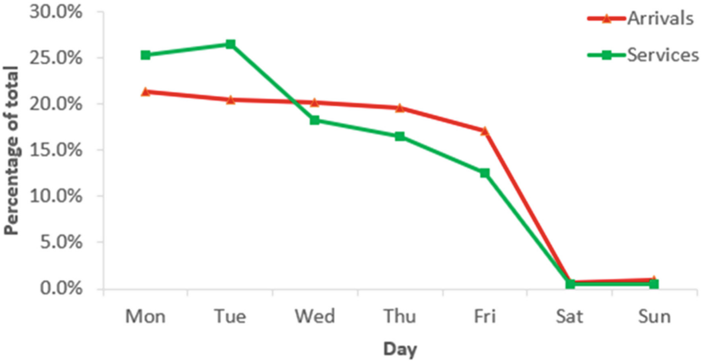


Fig. 8. Cumulative PBI arrival and resolution day percentages.

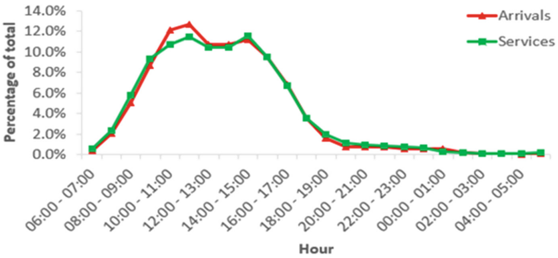


Fig. 9. Cumulative PBI arrival and resolution hours percentages.

Overall, most work done to create or resolve a PBI is done in relatively reasonable working hours. Given individual working preferences and time flexibility we chose 6am to 6pm UTC Monday to Friday as a reasonable period for a team based in Europe to open or close a PBI in a long-term sustainable, but flexible, way of working. The data shows that 92.8% of PBIs were created in this period and 91.8% were resolved in this period.

Figure 10 shows the distributions of PBIs created and resolved during potentially Unsustainable Hours. This shows that although the average Jira project had 9.3% of its PBIs created overnight or during the weekend and 11.4% of its PBIs resolved during this period, these averages are skewed by high concentrations at either end of the distributions. 19 Jira Projects, 6.4% of the total, had no Unsustainable Hours, and 18 Jira projects, 6.1% of the total, had more than one-fifth of their PBIs created and resolved during Unsustainable Hours. For this latter group, it is possible that the team was working for a client based outside a European time zone, or that the client required out-of-hours deployments. It is also possible that the team developed a meeting culture, forcing some work late at night, or that the sheer volume of work left the team feeling the need to work late. Of course, it is also possible that some individuals simply prefer to work unusual hours. Since we cannot discount any factor, we have included all the data to allow us to analyse the relationship between the Stability Metric, Inventory Days, and these Unsustainable Hours percentages.

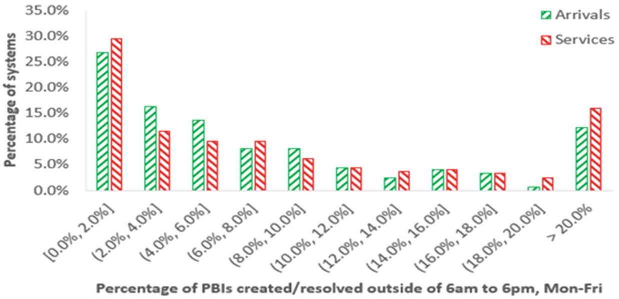


Fig. 10. Unsustainable Hours distributions for PBI creation and resolution.

4.3 Stability Metric, Inventory Days, and Unsustainable Hours

Figure 11 shows that there is essentially no correlation between the Stability Metric and Unsustainable Hours worked. The Spearman rank coefficient for created PBIs is 0.02 and for resolved PBIs is 0.04. This means that the flow of work in the system does not have any impact on the tendency for team members to work Unsustainable Hours on these Agile systems.

Figure 12 shows that there is also essentially no correlation between the Stability Metric and unsustainable hours worked. The Spearman rank coefficient value for created PBIs is -0.03 and for resolved PBIs is -0.05. This means that the volume of outstanding work does not have any impact on the tendency for team members to work Unsustainable Hours on these Agile systems.

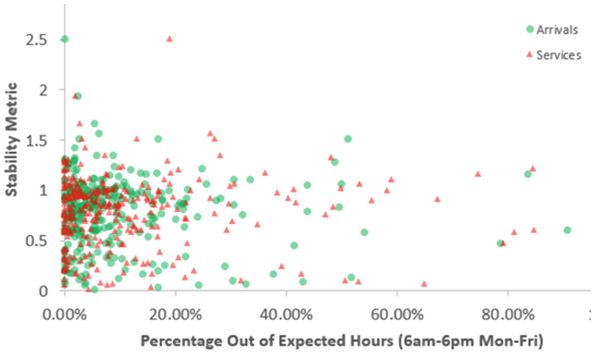


Fig. 11. Stability Metric versus Unsustainable Hours

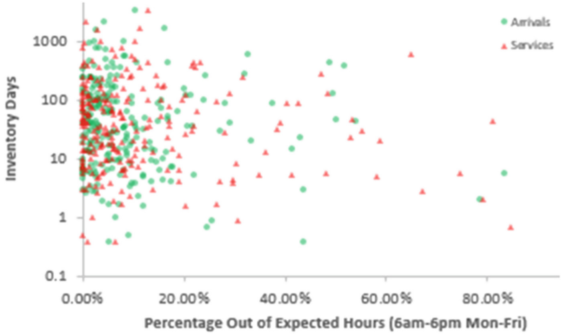


Fig. 12. Non-zero Inventory Days versus Unsustainable Hours

5 Discussion

The sustainability of work systems is of increasing importance with high economic costs to companies and health and well-being costs to workers. We analysed the tendency of workers to work late at night and over weekends, outside core working hours and then we compared this to systemic factors, such as the flow of work and the potential pressure of having a large amount of work to complete. Using data from real-world Agile teams in one company, we demonstrated these factors do not appear to be correlated.

This is an interesting and somewhat surprising finding as it suggests that whatever motivated the individuals in these projects to create or resolve a Jira PBI late in the night or on a Saturday, it was not the volume of work or the speed at which work needed to progress. There was as much tendency to work Unsustainable Hours when the team seemed to be running out of work as when there were piles of work to be done. While it might be understandable that a team member resolving a story or bug may not be aware of the size of the backlog or the number of new PBIs being created every day, it is less understandable how a Product Owner would not be somewhat aware of these things. Future work to introduce the Stability Metric and Inventory Days to real teams may be able to show if awareness of system metrics has an impact on behaviours.

In Sect. 2.1 we described two assumptions required to define Unsustainable Hours for these projects. While our data shows that working overnight or at the weekend did not appear to have an impact on system size or stability, we cannot say what impact this behaviour had on the individual, the team, or the quality of the work completed during unsustainable hours. Some people may simply prefer to work late at night. Future research is required to understand people's motivations to choose when to work and, at times, overwork unnecessarily.

Throughout this study, we treated Jira PBIs as being equal. The teams were likely to have assigned different levels of time-criticality to individual pieces of work. Although it is not presented above, we have measures of lead times for different issue types as these can be calculated from the difference between the time a PBI was resolved and the time it was created. We have ongoing work investigating patterns of lead times for different issue types that may help diagnose time-criticality as an extrinsic motivation for unsustainable hours. However, as Table 1 shows, the primary issue type in this dataset was the Bug with 49,525 PBIs. Since some bugs are likely to be more critical than others, and it is not impossible that a story or other PBI may become critical, other datasets or research will be required to investigate this factor of working late nights. The next section presents some of the limitations of this study in detail.

6 Limitations

Compared to previous larger datasets examined, the Closed Jira Projects Dataset had the advantage of better contextual realism and a higher confidence that these Jira Projects were Agile in nature. However, to protect confidentiality only historic closed projects were shared, and only process data. Some of the projects had teams migrate to other systems, usually client systems, and in other cases, the project came to an end for commercial reasons. This limitation means that the results may be somewhat skewed toward having lower inventory days than active projects. For example, 19.8% of all closed projects had all of their then outstanding PBIs marked as closed on their final day. Future work to measure stability and inventory days on active Agile teams would be useful.

A second potential benefit of repeating a similar study in active Agile teams would be to measure the motivations for working at unexpected times. Because the data came from many projects over a 15-year period, it is likely that most common scenarios were at some time encountered but we cannot assert that for certain. Other studies of either many Agile teams or more detailed longitudinal studies of individual teams may be able to discount extrinsic systematic factors for late nights or weekends and separate those from intrinsic motivations where a person is inspired to sacrifice rest for a solution.

A limitation of this study was our use of the period between 6pm to 6am Monday to Friday to denote "Unsustainable Hours". Figure 9 shows that over 90% of work performed was outside of these hours. However, it is plausible that individuals may prefer to work inside of these times. Also, anyone who worked a weekday between 6am and 6pm would regularly be working a 60-h week which would not appear to be sustainable. Future work with active Agile teams should ask individuals when they would prefer to work before measuring when they do work as an improved method of measuring true excess work.

This study used Jira records as a proxy for work performed. While creating and resolving PBIs could not be classed as a leisure activity we cannot be sure to the degree that one team member stayed up until 3 am to correct a tricky bug compared to another person deciding to do some minor Jira administration having woken at 3 am to let the cat out. Both will appear in our records the same, but the first person may feel tired from work while the second is likely well-rested. This study is limited to the accuracy of the data coming from a large number of teams across a range of projects, but all working in a single company.

7 Conclusion

Overwork is a dangerous activity that has increased over the last twenty years that has negative impacts on workers' physical and mental health. Agile is an approach to work that has become popular over the same period. There is a need to consider backlogs and other forms of partially completed work as a form of inventory and a potential source of waste. This study examined if the volume or the speed of flows of work related to the number of out-of-hours work being performed. We found no evidence to suggest that either the speed or volume of work was related to excess hours. This finding may be useful to other researchers seeking to examine the actual causes of excess work. There were Jira projects that demonstrated some excess work. The variation of the projects in time, type, and duration meant it was unlikely that the causes of what kept individuals working late at night and over weekends were common. This means that while some of the motivation to work late may be extrinsic, some appear to be intrinsic. This research presents an opportunity to repeat these procedures with active Agile teams to survey participants as to their preferred work behaviours as well as to investigate the advantages and disadvantages of having constant access to work systems.

Acknowledgements. The authors would like to acknowledge the contribution of intive in providing anonymised process data from historical projects.

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 to Lero—the Irish Software Research Centre.

Disclosure of Interests. Robert Healy is a past employee of intive. Edwin Lewzey is a current employee of intive. All other authors have no competing interests to declare that are relevant to the content of this article.

References

1. Mental health at work. https://www.who.int//news-room/fact-sheets/detail/mental-health-at-work/. Accessed 29 Jan 2024
2. Goal 8: Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all. https://sdgs.un.org/goals/goal8. Accessed 29 Jan 2024

3. Long working hours increasing deaths from heart disease and stroke: WHO, ILO. https://www.who.int/news/item/17-05-2021-long-working-hours-increasing-deaths-from-heart-disease-and-stroke-who-ilo. Accessed 29 Jan 2024
4. Fowler, M., Highsmith, J.: The agile manifesto. Softw. Dev. **9**(8), 28–35 (2001)
5. Healy, R., Dey, T., Conboy, K., Fitzgerald, B.: A novel technique to assess agile systems for stability. In: Stettina, C.J., Garbajosa, J., Kruchten, P. (eds.) XP 2023, vol. 475, pp. 20–33. Springer, Cham. (2023). https://doi.org/10.1007/978-3-031-33976-9_2
6. Beecham, S., Noll, J., Richardson, I.: Using agile practices to solve global software development problems -- a case study. In: 2014 IEEE International Conference on Global Software Engineering Workshops, Shanghai, China, pp. 5–10 (2014). https://doi.org/10.1109/ICGSEW.2014.7
7. Rusconi, C.: Sustainability Aspects inside the Agile Framework. Master thesis, MSc Sustainable Entrepreneurship University of Groningen, Faculty of Economics and Business (2020)
8. Hoda, R., Noble, J., Marshall, S.: Balancing acts: walking the Agile tightrope, pp. 5–12. Association for Computing Machinery, New York (2010)
9. van Oorschot, K E., Sengupta, K., van Wassenhove, L.N.: Under pressure: the effects of iteration lengths on agile software development performance. Proj. Manag. J. **49**, 78–102 (2018). https://doi.org/10.1177/8756972818802714
10. intive. https://intive.com/. Accessed 29 Jan 2024
11. At intive, We All Work Hybrid!. https://intive.com/careers/at-intive-we-all-work-hybrid. Accessed 29 Jan 2024
12. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit, pp. 4–8 Addison-Wesley, Boston(2003)
13. Krajewski, L., Ritzman, L., Malholtra, M.: Operations Management: Processes and Supply Chains, 9th edn. Pearson Education, Boston (2010)
14. Inventory Days Formula: Calculating Inventory Days. https://www.bluecart.com/blog/inventory-days. Accessed 29 Jan 2024

Onboarding for an Agile Software Development Company

Tomi Enberg[1](✉), Sari Alander[1], and Maria Paasivaara[2]

[1] LUT University, Lappeenranta, Finland
tomi.enberg@gmail.com
[2] LUT University, Lahti, Finland
Maria.Paasivaara@lut.fi

Abstract. The global shortage of highly skilled employees has created a need for a more efficient onboarding process in software development companies. The experienced onboarding efficiency affects the new employee's willingness to stay within the company and the efficiency of their work. Our case company transferred to a team-based organization structure, where agile teams were given high autonomy. Supervisor responsibilities were divided among different roles, such as people coaches and a lead team. We studied the onboarding process through a survey that was answered by 39 persons, as well as 18 interviews to gather details from the onboarding process. To validate the findings, workshops, and presentations were organized in the case company. We found that mentoring was experienced as the most important form of onboarding in the case company. The biggest challenges were lack of transparency, insufficient material availability, and lack of documentation. As solutions were suggested: a shared repository of onboarding materials, checklists for teams and mentors for onboarding new employees efficiently, and collecting team-specific materials to a single location.

1 Introduction

In technology companies worldwide, the need for new hires is so high that it threatens to limit the growth of many industries, one of which is the software industry [4,11]. The shortage of employees is partly due to lacking education and training and the ongoing technology boom due to the Covid-19 pandemic and the need is not only for employees in general, but for especially highly skilled and specialized workers [4,11,15,24].

To be able to acquire employees with a specialized skill set, a company must either hire such an employee, train one from its own pool of employees, or train a new employee using a trainee program. The limited number of employees in the industry makes recruitment difficult, costly, and even risky. According to a recruitment company SmartRecruiters [18], an efficient recruitment process requires 15 steps, which does not even include the onboarding steps of the newly hired employee. Onboarding of newcomers most commonly relies on a process, in which the newcomer learns the ropes of his or her new assignment [1,20].

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 144–162, 2024.
https://doi.org/10.1007/978-3-031-61154-4_10

This adds the expenses of the recruitment process [6,16,21]. However, not all companies see the importance of onboarding and consider it as a "necessary evil", to which not all teams or employees want to contribute [16].

As a case organization for this study, we chose a software company that constantly recruits new software engineers, and has recently recognized the challenge of not being able to provide a consistent onboarding experience for all new employees. Organizational changes have led to gaps in the process, where ownership and responsibilities of onboarding newcomers are unclear. The major changes include a change in management, the sudden shift to remote work induced by the Covid-19 pandemic, and the focus on implementing agile practices in all functions. Only limited research exist on how the agile ways of working could be efficiently combined with onboarding processes and practices [10]. In this paper, we study the challenges and successes of onboarding in an agile case organization and add to the body of knowled of agile onboarding.

2 Background

2.1 Onboarding

According to Bauer "Onboarding is the process of helping new hires adjust to social and performance aspects of their new jobs quickly and smoothly" [1]. A lot of research on the subject uses the term "Organizational socialization" [2,8,20]. In more recent research, organizational socialization is more commonly referred to as onboarding [1,7,17].

Generally, onboarding has been studied extensively. Whenever a person starts in a position, whether it is in a new company or a new position within the same company, there is always a period of socialization [20]. A person going through that period will be in an anxiety-producing situation and to reduce their anxiety, the newcomers will focus on learning the requirements of their new role as quickly as possible [19]. The more formal the process is, the more stress there is for the newcomer [19].

Employees, regardless of the field of industry, do change jobs and that includes challenges for both the individuals and companies [6]. Good onboarding – or organizational socialization – will have a higher chance to keep the new employees in the organization for longer than people who experience ineffective onboarding [2].

2.2 Theoretical Background

Van Maanen and Schein's Model: As Van Maanen and Schein stated, there was no formal model to study or understand onboarding prior their work [20]. Van Maanen introduced seven dimensions which describe the major strategies used to onboard people to the organization [19]. Van Maanen and Schein further developed this theory and defined six major tactics [20]: 1) Collective vs. individual onboarding processes, 2) Formal vs. informal onboarding processes, 3)

Sequential vs. random steps in the socialization processes, 4) Fixed vs. variable socialization processes, 5) Serial vs. disjunctive socialization processes, and 6) Investiture vs. divestiture socialization processes.

Jones Classification: Jones [12] presented the dimensions initially presented by Van Maanen and Schein [20] in a table with classification. Jones emphasized that the tactics used by organizations have an impact on the newcomer orientation and need to be understood better. Jones studied the actual response the different tactics may have on individuals, which provides an initial empirical standpoint on the different tactics used by the companies. The classification helps to explain Van Maanen and Schein's [20] dimensions as tactics that are either institutionalized or individualized. The main findings from the study by Jones, confirm their hypotheses that self-efficacy will be more influential than whether the tactics done by employers are institutionalized or individualized [12].

Bauer's Onboarding Metrics : Bauer [1] presents a model of onboarding functions and practices along with success measures. On an individual level, the building blocks for successful onboarding are four C's: Compliance, Clarification, Culture and Connection. As Bauer explains, most firms can be put on a scale of three levels, using the four C's: Passive onboarding, high potential onboarding and proactive onboarding. These can be used to define the current level of onboarding within a company.

2.3 Previous Research on Onboarding Within Software Development Companies

Even though onboarding in general has been studied extensively, we found only five papers on onboarding in software companies, two of them concentrating on agile [7,10].

Britto et al. studied the effects of having multiple software development teams scattered in different locations and contextualised their findings in a later study [5,6]. They found clear aspects that made the onboarding successful for new employees [5]: providing clear expectations for the new employee, providing extensive coaching and support on learning the project, and using tools to gather and give feedback. In their later study they found the challenges that had a negative influence on the new employee adjustment: distance to mentors, too formal onboarding process, too large tasks during onboarding and team instability [6].

Gregory et al. studied recently a team working in an agile setting [10] and used the onboarding model by Bauer to expand on [1]. The model created by Gregory et al. outlines two distinct aspects that no prior paper has considered. The first are the workplace adjustments, which are introduced as a new category in the model. The second is the realization of agile practices helping newcomers adjust to the company. The workplace adjustments are done by the company to enable better support for the newcomer [10]. These are team composition, team communication and the communities of practice.

Team composition refers to the adjustments the team has to consider when onboarding new employees. As agile software development focuses on the team-based decision making and quick reactions to changes, higher team cohesion and motivation leads to a more efficient team [23]. Introducing a new team member means that the team must re-adjust to accommodate a newcomer [10]. Gregory et al. explained that continuous personnel changes within the team limited the team's ability to form into a cohesive agile team, which in turn did not allow time for anyone to start mentoring. When the team was able to form into a balanced team, the newcomers could be better accommodated to become a part of the team, with the help of a mentor.

Team Communication: Agile teams that have been working together for a longer time, may have developed a style of communication which is difficult to understand by the new employees [10]. Accommodating newcomers requires the more experienced team members to consider newcomers' level of understanding of the domain and terminology.

Communities of Practice: Any agile team is also a part of an organization. The organization can have similar teams or areas of expertise that give the team – and most importantly the newcomer – the feeling that they can share their knowledge and gain understanding from their organization, not only from the team [10]. This shared expertise in the organization is known as "communities of practice" [22].

In another study done in agile settings Buchan et al. [7] provided best practices to focus on while onboarding: socialization opportunities, access to high quality knowledge artefacts, access to formal training, proactive feedback and knowledge sharing and providing psychological safety to experiment and learn.

Most recently, Rodeghero et al. [16] studied the remote onboarding of software developers during the pandemic at Microsoft. They found that social interactions within the teams were lacking in depth. Most employees felt connected to their team, but some employees felt the team activities lacking. The best practices the study discovered were team activities, such as game nights, hackathons and recurring meetings to just chat and catch up. The challenges found included not seeing other people's faces, as cameras were commonly turned off, and asking for help and connecting with the team were considered difficult [16]. The authors provided a list of adjustments to be done by the company to help with onboarding new employees more efficiently: 1) promote communication and asking for help, 2) encourage teams to turn cameras on, 3) schedule 1 to 1 meetings, 4) provide information about the organization, 5) emphasize team building, 5) assign an onboarding buddy, 6) assign an onboarding technical mentor, 7) support multiple onboarding speeds, 8) assign a simple first task, 9) provide up-to-date documentation.

2.4 How to be Efficient When Onboarding People in an Agile Setting?

When looking at research done with agile teams, it seems that mentoring and peer support are common and suggested techniques for onboarding new employees to agile teams [7,10] Along with mentoring, agile practices and ways-of-working also function as onboarding practices, for example sprint retrospectives [10]. According to Cockburn [9] agile companies tend to lean towards a leadership strategy that assigns goals and communicates constraints, rather than controls how work is done within the team. Agile teams are self-organizing and have a lot autonomy, thus no model, process or tool can be imposed to an agile team from higher up. In a modern, agile environment, teams take into use tools, techniques, and methods they find useful. For effective process improvemet, agile teams must want to use that process, method or tool.

Using the Van Maanen and Schein's [20] dimensions, the agile practices could be summarised requiring the following dimensions: individual, informal, sequential, fixed, serial and investiture. When comparing these to Jones' classification, it seems that the most efficient way to teach the job context would be using individualised tactics [12]. This is due to the teams having the freedom to choose their own processes within certain frames and thus also find the best ways for them to onboard new employees [3,10].

The content of the job on the other hand should be clear for all employees from the start. According to previous studies, there might be challenges when it comes to handling expectations and learning [10,16]. Also, Jones stated that random and variable tactics bring unnecessary uncertainty for the new employee [12].

Finally, socialization seems to be important. Not having connection to the members of your own team, makes asking help more difficult [16]. Not being able to onboard into an organization socially may be a big factor when the new employee is making the decision to either staying in the company or leaving [2].

3 Research Method

3.1 Research Objectives

This paper aims to find out the challenges and solutions for onboarding new employees in an agile software product organization. The findings can help other companies with similar challenges as our case organization was experiencing to overcome issues and as Bauer mentioned [1], increase the employee retention, employee well-being and effectiveness in the company. In this paper we aim to answer the following research questions:

RQ1: What are the experienced challenges when onboarding new employees into an agile team?

RQ2: Which practices support onboarding new employees into an agile team?

RQ3: How to improve the onboarding process for agile teams within a company?

To answer the research questions above, we collected data by a survey (39 responses), interviews (18 interviewees) and workshops (10 participants). Next, each of these are described in detail. Before that we briefly describe the case organization.

3.2 Case Organization and Their Research Premise

The company under study operates in the software engineering industry. It employs over 350 people and has a revenue of over 80 million euros. The 350 people within the company work with their own assigned products, which are managed individually. Out of these individually managed product development teams, a product development unit of 82 people underwent an organizational transformation towards a team-based organizational structure. In 2020 the global pandemic moved people to work from home and many of them did not return fully to the office after the pandemic, but continued in hybrid mode. The data collection of this study took place in 2022, after the company had moved to hybrid work mode. Agile methods, such as Scrum, are widely used within the company, by different teams. Each team works mainly independently, specializing in its specific area of the product. The teams are self-sustaining and self-directing, and may choose their own practices and processes within certain frames. This applies for onboarding as well. The company continuously hires new people to support the growth of its products.

After the transformation new roles were introduced and responsibilities were re-arranged to reduce person dependence. The onboarding responsibilities of supervisors and directors were reduced, and new lower-level roles were introduced. Many of the roles are considered as "hats", which are additional responsibilities on top of an assigned role. For example, a person can be both a developer and a Scrum Master, or a Product Owner and a people coach. The new roles central to onboarding are the following:

People Coach: A person responsible for long-term career coaching, probationary period follow-up, goal setting and well-being of a predetermined number of employees within the R&D. The employees may choose the most suitable people coach for themselves. The objective is to provide a peer for each employee that ensures that the employee has the conditions to succeed in his or her work assignments. A people coach usually comes from a different team. This allows for the employee to receive ideas from outside the own team, and to talk about the situation in the own team more openly.

Mentor: A team colleague from the own team that is responsible for providing a basic understanding of the team processes and practices, along with the onboarding in the team, including the specifics of the development or design environment. Mentors have usually dedicated time to answer questions the new employee might have.

At the time of the study, employees were onboarded to their assignments with the use of a team mentor and a people coach, along with general company level onboarding. The onboarding has been a challenge for many years, as tools and technologies kept changing and no sufficient documentation was always available. According to the statistics the company has collected, onboarding does have a positive impact, even though it is sometimes criticized of being insufficient. This may be due to the development teams not having a culture to share their practices and common materials. The process of onboarding has become somewhat blurry for both the company and the new employees. Due to these challenges, research was welcomed to advance the process around onboarding to support the people coaches, mentors and new employees to adjust to the company better and with better consistency.

3.3 Scope and Perspective of the Study

To limit the scope of this study, we chose to concentrate on the one product organization with approximately 80 people, with the most changes in ways of working. The data was collected by the first author of this paper, who at the beginning of this study had worked as a developer in the company for two years in the R&D of the product under study. While starting the study, the author was transferred from the development team to concentrate on improving the onboarding process and to additionally work as a people coach for summer trainees. To avoid close involvement into the onboarding process of the summer trainees, the summer trainees in question were excluded from this study. Even though the close involvement of the first author can be seen as a limitation, the people coach role provided also good insight into the onboarding process. The two other authors participated in designing and monitoring the study steps, ensuring the the objectivity. They provided an outside view, as well as contributed to the writing of this paper.

3.4 Data Collection

Survey Data Collection: The initial status of the current situation of the onboarding processes was collected with a survey. The survey questions were formed by drawing inspiration from previous studies [1,14]. As onboarding consists of multiple stages as presented by Bauer [1], the survey needed to be tailored to gather a wide perspective on the onboarding in the case company. Questions[1] asked about general onboarding experience, the support received, the knowledge of the company, and the preferred ways to receive information while onboarding. The survey consisted of two questions about the respondent's backgrounds, 13 agree-disagree questions on a likert scale [13] about the onboarding, and five open-ended questions in which the respondents were asked to describe the challenges and good practices of their onboarding experience. Respondents could also indicate if they would be willing to participate in the interviews on the topic.

[1] https://figshare.com/s/8b90fd8cc32335f0bbad.

The questions were reviewed by employees in the case company and all authors of this paper and improved based on the feedback. Two employees tested the survey and it was estimated that responding would take 10–15 min.

Link to the survey was sent to a common mailing list of the product's R&D. The 82 recipients consisted of developers, designers, quality assurance specialists, Product Owners, and lead experts. We received 39 completed responses (response rate: 47,6%).

Interview Data Collection: Semi-structured interviews were selected to collect more detailed description of the employees' onboarding experience, the same way as in the previous onboarding studies [10,14]. The interview questions were formed by analysing the previous literature on the topic (e.g. [14]) and modified to better reflect large-scale agile set-up in the case company. Moreover, the preliminary findings from the surveys were used to ask more specific questions. The interview questions can be found at.[2]

To get an extensive picture of all different onboarding processes, tools, or techniques in use across different teams, we found it important to have interviewees from all 13 teams. Initially, ten people indicated in the survey study of their willingness to be interviewed. They represented six teams. The organization chart and previous knowledge on employees' roles was used to find people that represented groups and roles that were not yet represented in the interviews. The purpose was to have a selection of employees that represented a wide range of roles and groups.

In total 18 people from 12 teams were interviewed. 14 interviews were conducted using the questions aimed specifically to ask about the interviewee's own onboarding experience, while in four interviews the main purpose was to gather information of the recent mentoring experiences from more experienced employees working as mentors. This separation was made to make it easier to focus on the interviewee's own point of view, in cases where the employee did not recall their own onboarding experience. Interviewees, their roles, teams and time from own onboarding are listed in Table 1.

Interviews were confidential, and it was made sure that in the results the respondents cannot be traced back to any team or a specific employee to ensure privacy.

Workshops: All 18 interviewees were invited to participate in an online workshop. Finally, a total of 10 people participated in the workshop which was divided into two parts due to time constraints. Out of the total 10 participants, 9 participated in the first part of the workshop, and covered a total of 7 teams. These 9 people were divided into three break-out rooms to collect ideas and create solutions to the challenges identified in the interviews. When organising the second workshop, all the initial 18 interviewees were invited, out of which 6 participated in the second workshop, and covered a total of 5 teams. The focus was to create

[2] https://figshare.com/s/8b90fd8cc32335f0bbad.

Table 1. Interviewees

Interviewee ID	Role	Group	Team identifier
Interviewee 1	Developer	More than 2 years	1
Interviewee 2	Developer	More than 2 years	1
Interviewee 3	Developer	More than 2 years	2
Interviewee 4	Other R&D Employee	More than 2 years	1
Interviewee 5	Developer	Less than 1 year	1
Interviewee 6	Other R&D Employee	Less than 1 year	3
Interviewee 7	Other R&D Employee	More than 2 years	3
Interviewee 8	Developer	More than 2 years	4
Interviewee 9	Developer	More than 2 years	5
Interviewee 10	Developer	1–2 years	6
Interviewee 11	Developer	Less than 1 year	7
Interviewee 12	Developer	More than 2 years	8
Interviewee 13	Developer	Less than 1 year	9
Interviewee 14	Developer	1–2 years	10
Interviewee 15	Developer	More than 2 years	8
Interviewee 16	Developer	1–2 years	11
Interviewee 17	Designer	More than 2 years	12
Interviewee 18	Developer	1–2 years	1

concrete action points based on results from the survey, the previous workshop and the interviews.

Due to having workshop participants from multiple offices, an online whiteboard tool Miro was used as a working space for the workshop. Each group, based on the division in break-out rooms, had their own working space, in which the participants could create new sticky notes for their ideas, and use the sticky notes to brainstorm and prioritise their ideas. This helped the analysis, as all the workshop work was visible and written out in the online tool. The workshop participants were told to create solutions for the challenges they chose from the pool of challenges found previously in the study, as well as create concrete action points on how to approach solving the problems.

Data Analysis: In this paper, we concentrate on the data collected with the open-ended questions of the survey, interview data, and workshop results. Both survey and interview data were analysed using an Excel sheet where individual, good practices, challenges, development ideas, and other common thoughts were collected. A matrix was formed from the respondents and the key factors, from which it was easy to count the number of mentions that each key factor had. Thus, the most commonly mentioned challenges, good practices, and ideas could be easily identified. The end result of the workshops was ready-made solution suggestions.

4 Results

4.1 RQ1: What Are the Experienced Challenges When Onboarding New Employees into an Agile team?

The case organisation had been experiencing challenges regarding onboarding and therefore saw a need for this study. The challenges recognized in this study are listed in Table 2. The table shows how many employees brought up a certain issue in the open-ended survey answers and during the interviews. The order in the table is based on how many mentioned each challenge during the interviews. As many survey respondents volunteered for the interviews, there is some overlap between the survey and interview respondents.

Table 2. Challenges identified from surveys and interviews

Challenges	Description	# mentions in surveys	# mentions in interviews
C-1	Limited transparency of onboarding practices in the company	–	11
C-2	No onboarding feedback gathered within R&D	–	7
C-3	Materials or documentation missing	4	6
C-4	Organizational aspects unclear after initial presentations or experience	2	5
C-5	Unclear responsibilities with onboarding	5	5
C-6	Materials or documentation outdated	1	5
C-7	Common R&D meetings use terminology unknown for newcomers	–	3
C-8	Materials and documentation spread out to multiple systems and folders	3	3
C-9	Feeling of not included in the team	1	3
C-10	Onboarding events organised too late and using too advanced terminology	1	3
C-11	No knowledge on other team's responsibilities	–	3
C-12	Access restrictions to services and materials (on team and R&D level)	1	2
C-13	Lack of experise-area specific knowledge and support	–	2

There is a clear difference between the challenges brought up in the survey compared to the interviews. The survey answers emphazise the low quality and lack of onboarding materials and documentation, whereas interview brought up topics such as lack of transparency of onboarding practices and not collecting feedback on the onboarding experience.

A challenge with the highest number of mentions was surprisingly the *lack of transparency of the onboarding practices* (C-1). In the interviews, most interviewees mentioned having a dispersed process, which had some overlapping and missing items. The interviewed mentors mentioned that they had little or no understanding on the onboarding process as a whole and had to purely focus on the mentoring and onboarding of the team. Lack of transparency was also a problem when looking at the onboarding process between different teams, as the onboarding process was quite different for different teams (C-11). This lack of transparency was only visible in the interviews, as when people described their process and experiences it became clear that they had no knowledge of other teams or mutual practices.

No onboarding feedback was gathered (C-2) was the second most mentioned challenge. In the interviews, it became clear with almost all employees that no feedback was asked about their onboarding process. This makes the process development difficult, as there is no way of knowing what aspects to keep the same and what to improve. The different events organised jointly for different products within the company did have feedback models in place, but mentoring or team onboarding did not have any kind of feedback process. In the interviews, three people mentioned that the onboarding survey conducted during this study was the first time they had been asked for feedback regarding onboarding.

Missing materials or documentation (C-3) was an issue that came up both in the survey and in the interviews. Interviewed persons emphasized that missing materials and documentation was a huge issue and caused major delays in their onboarding. Additionally, documents were mentioned to be outdated (C-6) and people had to look for information from many places without having a clear picture of what to look for (C-8). In addition, two interviewees brought up that they found materials, but did not have access to those materials, and had to ask for them specifically (C-12). Also, one survey respondent mentioned not getting started for a few days, because he or she did not have access rights. A few interviewees mentioned that they did not lack materials, but they lacked a structure for the onboarding materials.

When it comes to the organization structure and internal language, there were mentions about the *organization structure being unclear* (C-4), not knowing other team's responsibilities (C-11), or not knowing the right people to ask for help when it comes to a more specific area (C-13). Moreover, some employees mentioned *not initially understanding the domain and business-specific language* (C-7). The interviewees mentioned these as smaller issues. However, five interviewees reporting of not understanding the organization structure would indicate that general organization and company-wide aspects have not been communicated well enough during the onboarding process.

One of the most worrying findings were the mentions of *not feeling included in the team* (C9). This had as many reasons as there were mentions and were mostly individual in nature. The underlying cause appeared to be the way teams to give specific responsibilities to their new team members. Most of these responsibilities were something no one had an ownership before, causing *the new employee to fall into a role, where no one could instruct him or her on what to do* or even on where to start (C-13). When being a part of the team, but still only completing certain assignments that were not related to the assignments of the team, caused that some employees were not properly included in the team's work. This is an indication of a wider problem, the lack ownership of responsibilities, also apparent in responsibilities related to onboarding itself (C-5).

4.2 RQ2: Which Practices Support Onboarding New Employees into an Agile Team?

The currently employed practices that are mentioned as successful in supporting onboarding in the case company can be found from Table 3 which is constructed based on the interview answers.

Table 3. Good onboarding practices identified

Practice	Description	# of mentions
P-1	Mentoring	11
P-2	People coaching	7
P-3	Encourage asking questions, arrange activities and facilitate situations for new employees to participate into conversations	7
P-4	Onboarding day(s) at the office	6
P-5	The team as a whole being open to questions from the new employee	6
P-6	Pair/mob/team programming	4
P-7	Trust in other functions and their practices without having visibility	3
P-8	Getting to do work assignments from the first days on	3
P-9	Actively giving the new employee freedom and opportunities to learn the practices and ways-of-working of the team and the assignment	3
P-10	Common R&D meetings offering insight into business and organizational aspects of the case organization	3
P-11	Ability to influence different processes even as a newcomer	3
P-12	Supervisor onboarding (showing product vision, strategy, organization working habits, and practical aspects of the job)	3
P-13	"Open camera/mic" – policy when working remotely	2
P-14	Getting to complete challenging and varying tasks from the start	2
P-15	Actively encouraging respectful behaviour in all contexts	2

The most mentioned practice both in the interviews and in the open questions of the survey was *mentoring* (P-1). Mentoring was described as one of the most important parts of onboarding, due to its focus on the individual and their skills. Mentors were always working in the same agile team. In a few responses the term “mentor” was not used, but the description was clearly the description of a mentor. The most important in mentoring was the availability of someone always ready to answer your questions. In addition, many people who had started prior to the pandemic mentioned that asking questions in the office from anyone was important and especially that *the team as a whole was open to questions from the new employee* (P-5). The pandemic stripped away the possibility of being at the office, so other ways to lower the threshold to ask questions were needed. In the beginning, it was harder for teams to find habits to help newcomers get help more easily, but the more recently onboarded interviewees mentioned different ways to *encourage asking questions, arrange activities and facilitate situations for new employees to participate into conversations*, which encourages to an open and transparent working culture (P-3). These include in different teams: 1) staying in the virtual daily meeting room after the daily activities with cameras open until lunch, 2) hanging out in a virtual meeting room with microphones open every day for at least a few hours (P-13), 3) having a weekly reservation on working with cameras open, 4) having a weekly time for just hanging out and talking about non-work-related stuff to build the team, 5) team gatherings and re-building the team working habits with the new employee, and 6) pair/mob/team programming (P-6).

The second most mentioned practice was *people coaching* (P-2). Due to the people coaching being a relatively new organizational model, only a few interviewees had a people coach from the start. The similar tasks as the people coaches were doing, were mentioned as being done by the previous supervisor more than two years ago (P-12). The supervisor welcomed new people to the office and showed them around. The interviewees onboarded less than two years ago on the other hand thought of the walk-through of the contract and strategy, when talking about supervisor onboarding.

Another good practice that was in use with many teams was *onboarding day or days at the office* (P-4). This is mainly something that was taken into use after the pandemic had started because previously people started their work at the office quite regularly. During the pandemic, the first day included presentations and setting up the environment for the work assignment. Unfortunately, not too many people were introduced with multiple days at the office as the regulations prevented working there most of the time. However, this was seen as important, due to getting to meet other people at the office and getting to know your teammates better. Onboarding days were also difficult for teams that had their members spread out to multiple offices.

Another thought that was mentioned multiple times in the interviews, was the hands-on work assignments given to new employees, especially *getting to do work assignments from the first days on* (P-8). Employees stated that they got familiar with the code by *getting to complete challenging and varying tasks* *from the start* (P-14). Their first assignments were one by one more challenging to solve, which was seen as motivating. Even the first assignments were actual backlog items from the team's backlog. A common thought here seems to be that the planning was done in advance. Along with planning the assignments in advance, reserving time for the newcomer to get to know the code without rushing was also thought as important, because it reduced the anxiety of meeting the performance expectations of a full-time employee right away (P-9).

4.3 RQ3: How to Improve the Onboarding Process for Agile Teams Within A company?

The third research question asks for solutions for improving the onboarding process in an agile organization. To find the best solutions, feedback, and solutions were collected first from the interviews. Additionally, a two-phase workshop was organized to facilitate ways for employees to solve the most important challenges by themselves.

Ideas Collected. Unlike previous categories, new ideas were something interviewees did not have much in common. We received a lot of new ideas, ranging from team-level onboarding ideas to product-level ideas. See Table 4 presenting the most often mentioned ideas. The idea, which many interviewees seemed to agree on, was a *clear presentation of the organization structure* (I-1). It would help all employees to better understand the different parts of the organization and their responsibilities.

Ideas that came up as successful and were already used in some teams were: *Pair/mob programming and completing assignments as one team* (I-2), which allows for information sharing during the regular programming, arranging a *team gathering* as soon as possible when a newcomer starts (I-3), *planning ahead when a newcomer is arriving* for example regarding joint office days and tasks for the newcomer (I-4), and that the *team will gather their tools, access rights and documents into a single location* for the newcomer and keeps those up-to-date (I-6). These ideas could be spread and taken into use also in other teams.

At the time of the study the onboarding responsibility was unclear and scattered between different people: people coach, mentor, team and the rest of the organization. As team is central for the onboarding success, many onboarding practices depend on the team and might be team-specific due to the self-organizing nature of an agile team, it was suggested that in the future the *team would take the main responsibility of onboarding* (I-5).

Finally, as teams have many things in common, it was suggested to create *common software architectural solutions accross teams* (I-7) and *common presentations explaining higher level architecture and ways of working in the organisation* (I-8). These would give more insight into the product and the organization, and make it easier for employees to understand the architecture and business decisions behind the product.

Table 4. Ideas collected

Ideas	Description	# of mentions
I-1	Clear presentation of the organisation structure	4
I-2	Pair/mob programming and completing assignments as one team	4
I-3	Team gathering as soon as possible when newcomer starts	3
I-4	Better planning ahead when a newcomer is arriving	3
I-5	Team will in the future take the main responsibility of the onboarding	3
I-6	Team will gather their tools, access rights and documents them to a single location for the newcomer. Team keeps those up-to-date	3
I-7	Common software architectural solutions accross teams (as applicable)	2
I-8	Common presentations explaining higher level architecture and ways of working in the organisation	2

Solutions Created in the Workshops: The workshop participants selected the following five challenges as the most important to solve in the onboarding process and offered a number of solutions for each challenge, as shown below:

Mentor Lacks Instructions: Each team should create their own *checklist for their own onboarding process*, so that the most essential technical details will always be reviewed. The checklist could contain: UI features, project structures, microservices, repositories and the most important data structures. The checklist requires checking and updating every time a new employee is starting.

Materials or Documentation Outdated: To avoid this from happening in the future, *links should always be updated* when documentation is updated. Having all documentation in one platform would support this.

Materials or Documentation Missing: All materials need to be found from a *single location.* Outdated material needs to be removed if found. A common repository for technical documentation already exists, but is not utilized. A wider adoption for this tool is needed. This would centralize the technical documentation.

Lack of Knowledge in Some Specific Domain Areas: The issue that the new employee does not know anyone with knowledge about a specific area, causes a lot of questions and time to find a person to answer a question. To solve this, *an introduction of the people*, and their skills and responsibilities also outside the own team would be needed.

Feeling of Not Beloning to the Team: More care should be taken *to onboard the new employee to the team* as well. This includes for example some activities with the team, preferably face-to-face. Some off-topic talks within the team can also help with team building. Team members should focus on including the more quiet people in discussions.

As can be seen from the list above, the solutions suggested by the workshop participants are quite low level and should be easy to implement.

The workshop participants were asked to form action points to help the organization to take suggestions into practice. Two major action poins were created: *1) Create a group to discuss onboarding best practices and challenges from different teams.* This would allow for knowledge and experience sharing between the teams about onboarding. This way the practices used in different teams could be made more common and generally more would be known about the onboarding within the organization. *2) Create a common checklist for onboarding and place for onboarding documentation.* The checklist could include also technical details that would be essential to know for a new employee. That way the mentor would have a clear place for all the materials, and the team could use the documentation for their own ways of working. It was also heavily suggested that this sort of a documentation would be in the version control, next to the code made by the teams.

5 Discussion and Conclusions

In this paper, we reported a case study on onboarding in a medium-size agile software development company, which had undergone changes and expressed the need to improve its onboarding practices. With a survey, interviews, and workshops we collected data on the current onboarding challenges, successful practices, and improvement ideas from the software development organization.

Even though there exists plenty of literature on onboarding in general, including onboarding theories [1,12,20], onboarding in agile software development has not received much attention yet. According to Gregory et al. [10], more research is needed to cover different kinds of agile companies, including remotely working and larger software companies, both of which were covered in this research. Next, we will discuss our results, especially from the point of view of agile software development.

Based on our results we present the following practical implications for onboarding in agile software development organizations:

1) **Mentoring is the key.** In our results, mentoring was the most often mentioned successful onboarding practice. A peer from the same team where the new employee would start was assigned as a mentor for him or her. When comparing to the literature, it seems clear that mentoring is widely used as a good onboarding strategy, even across different industries. In several software companies, mentoring was reported to be one of the most popular or the prime onboarding method, [5,7,10,16].
2) **Give Agile teams the main onboarding responsibility and make the responsibility clear.** According to our results, the agile teams had already a lot of responsibility in onboarding the new team members. However, that responsibility was not clearly stated. In agile software development the teams are self-organized, and according to agile leadership strategy teams are just

assigned goals and given constraints, rather than controlled by how the work is done within the team [9]. Therefore, it seems quite natural that teams have a lot of autonomy also in the onboarding process when including new members in an agile team.

3) **Give agile teams the autonomy to decide the onboarding practices for them.** Many of the successful practices found in our study, as well as improvement ideas were team-level practices, which implies how important the team-level practices are for successful onboarding. As agile teams have the autonomy to decide their internal practices, teams should be given the freedom to decide the best onboarding practices for them. Some of the successful and suggested practices in our study were actually typical agile practices, such as mob/peer programming. Previous literature has also reported that agile practices and ways of working also function as onboarding practices, for example, sprint retrospectives [10] and code reviews [5,10].
4) **Support and encourage the teams to share good practices.** Our results revealed that different agile teams had already a good selection of successful onboarding practices at the team level. However, the teams were not aware of the different practices used by different teams. Sharing these practices and learning from other teams was one of the actions suggested by the workshop participants. Previous research supports this finding, e.g., Gregory et al. [10] suggest communities of practice to share expertise in the organization in the form of "communities of practice" [22].
5) **Create a common place for onboarding documentation and checklists. Keep them up to date.** The lacking, missing, and not up-to-date materials for onboarding, such as plans and orientation materials, were brought up as a challenge in our study, and a lot of ideas were given on how to improve the situation. Also, previous research in software companies has reported similar problems [5,10,14]. These studies have suggested that better documentation would fill in many of the challenges faced by the new employees.

We call for more research on onboarding practices used in different agile development companies. In addition, it would be interesting to follow up on the influences of the improvements suggested by our study participants, and how these improvements might influence the employee engagement statistics and turnover rate in a longer period. Additionally, information and reports from implemented methods by each team would provide excellent insight into the effect the solutions have on the team's daily work.

References

1. Bauer, T.: Onboarding new employees: Maximizing Success (2010). https://teachercentricity.com/wp-content/uploads/2014/08/SHRM-Onboarding-Report.pdf
2. Bauer, T., Erdogan, B.: Organizational socialization: the effective onboarding of new employees. In: APA Handbook of Industrial and Organizational Psychology, vol. 3, pp. 51–64 (2011). https://doi.org/10.1037/12171-002

3. Beck, K., et al.: Manifesto for agile software development (2001). http://www.agilemanifesto.org/
4. Breaux, T., Moritz, J.: The 2021 software developer shortage is coming. Commun. ACM **64**(7), 39–41 (2021). https://doi.org/10.1145/3440753
5. Britto, R., Cruzes, D.S., Smite, D., Sablis, A.: Onboarding software developers and teams in three globally distributed legacy projects: a multi-case study. J. Softw. Evol. Process **30**(4), e1921 (2018). https://doi.org/10.1002/smr.1921. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1921
6. Britto, R., Smite, D., Damm, L.O., Börstler, J.: Evaluating and strategizing the onboarding of software developers in large-scale globally distributed projects. J. Syst. Softw. **169**, 110699 (2020). https://doi.org/10.1016/j.jss.2020.110699
7. Buchan, J., MacDonell, S., Yang, J.: Effective team onboarding in agile software development: techniques and goals, pp. 1–11 (2019). https://doi.org/10.1109/ESEM.2019.8870189
8. Chao, G.T., O'Leary-Kelly, A.M., Wolf, S., Klein, H.J., Gardner, P.D.: Organizational socialization: its content and consequences. J. Appl. Psychol. **79**(5), 730–743 (1994). https://doi.org/10.1037/0021-9010.79.5.730
9. Cockburn, A., Highsmith, J.: Agile software development, the people factor. Computer **34**(11), 131–133 (2001). https://doi.org/10.1109/2.963450
10. Gregory, P., Strode, D.E., Sharp, H., Barroca, L.: An onboarding model for integrating newcomers into agile project teams. Inf. Softw. Technol. **143**, 106792 (2022). https://doi.org/10.1016/j.infsof.2021.106792. https://www.sciencedirect.com/science/article/pii/S0950584921002329
11. Hyrynsalmi, S.M., Rantanen, M.M., Hyrynsalmi, S.: The war for talent in software business-how are finnish software companies perceiving and coping with the labor shortage? In: 2021 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), pp. 1–10. IEEE (2021)
12. Jones, G.R.: Socialization tactics, self-efficacy, and newcomers' adjustments to organizations. Acad. Manag. J. **29**(2), 262–279 (1986). https://doi.org/10.2307/256188
13. Joshi, A., Kale, S., Chandel, S., Pal, D.K.: Likert scale: explored and explained. Brit. J. Appl. Sci. Technol. **7**(4), 396 (2015)
14. Pavlina, K.: Assessing best practices for the virtual onboarding of new hires in the technology industry (2020). https://www.proquest.com/openview/39c7b814b8bf1d73adcabe97ed085ba4/1?pq-origsite=gscholar&cbl=44156
15. Rieff, J., Peschner, J.: Employment and social developments in Europe (2020)
16. Rodeghero, P., Zimmermann, T., Houck, B., Ford, D.: Please turn your cameras on: remote onboarding of software developers during a pandemic. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 41–50. IEEE (2021)
17. Sharma, G.G., Stol, K.J.: Exploring onboarding success, organizational fit, and turnover intention of software professionals. J. Syst. Softw. **159**, 110442 (2020). https://doi.org/10.1016/j.jss.2019.110442. https://www.sciencedirect.com/science/article/pii/S016412121930216X
18. SmartRecruiters: Hiring process steps for 2022 (2018). https://www.smartrecruiters.com/resources/glossary/hiring-process-steps/
19. Van Maanen, J.: People processing: strategies of organizational socialization. Organ. Dyn. **7**(1), 18–36 (1978). https://doi.org/10.1016/0090-2616(78)90032-3. https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=5142946&site=ehost-live

20. Van Maanen, J.E., Schein, E.H.: Toward a theory of organizational socialization (1977)
21. Vierhauser, M., Rabiser, R., Grünbacher, P.: A case study on testing, commissioning, and operation of very-large-scale software systems. In: Companion Proceedings of the 36th International Conference on Software Engineering, pp. 125–134 (2014)
22. Wenger, E.: Communities of practice: a brief introduction (2011). https://scholarsbank.uoregon.edu/xmlui/handle/1794/11736
23. Whitworth, E., Biddle, R.: Motivation and cohesion in agile teams. In: Concas, G., Damiani, E., Scotto, M., Succi, G. (eds.) XP 2007, vol. 4536, pp. 62–69. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73101-6_9
24. Wright, B.: Employment, trends, and training in information technology. OCCUP. Outlook Q. **53**(1), 34–41 (2009)

Exploring Human-AI Collaboration in Agile: Customised LLM Meeting Assistants

Beatriz Cabrero-Daniel[1](✉), Tomas Herda[2], Victoria Pichler[2], and Martin Eder[2]

[1] University of Gothenburg, Gothenburg, Sweden
beatriz.cabrero-daniel@gu.se
[2] Austrian Post, Vienna, Austria

Abstract. This action research study focuses on the integration of "AI assistants" in two Agile software development meetings: the Daily Scrum and a feature refinement, a planning meeting that is part of an in-house Scaled Agile framework. We discuss the critical drivers of success, and establish a link between the use of AI and team collaboration dynamics. We conclude with a list of lessons learnt during the interventions in an industrial context, and provide a assessment checklist for companies and teams to reflect on their readiness level. This paper is thus a road-map to facilitate the integration of AI tools in Agile setups.

Keywords: Agile · Scrum · meetings · human-AI collaboration

1 Introduction

Team collaboration and meetings are an essential part of any software development organisation, but they are challenging to organise and manage. Often, meetings do not follow guidelines or run into issues that affect their efficiency and productivity, or delay decision-making [12]. Sometimes, the guidelines themselves burden the development teams and need to be adapted. The Post-Rolling Refinement Model (PRIME), an in-house designed Scaled Agile framework proposed by the *Austrian Post* [14], aims to reduce this burden.

Microsoft sparked in 2019 the debate on how Artificial Intelligence (AI) could further improve meetings by automating tasks and retrieving information before, during, and after them [10]. That could, among other benefits, improve the flow, save time, increase productivity, or reduce frustration during said meetings [10]. There is a growing body of white and grey literature that recognises the role that AI could have in reducing the organisational burden on the participants, ensuring that meetings are conducted in a more organised and structured manner, or providing insights to improve future meetings. However, there has been

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-61154-4_11.

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 163–178, 2024.
https://doi.org/10.1007/978-3-031-61154-4_11

little academic attention paid to the role of AI interventions in software development meetings. And, to the best of the authors' knowledge, no research has been conducted to investigate the support of an AI meeting assistant and its interactions with practitioners in a systematic way.

We are adopting action research to explore the use of AI in meetings at *Austrian Post Group IT*, an international postal, logistics and service provider described in detail in Sect. 3. We focus on two of their regular meetings: standard Daily Scrums and feature refinement and planning meetings, part of the PRIME framework [14]. The study explores how the use of AI affects the practitioners' meeting experience. By doing so we aim to answer three questions:

RQ1 How can AI assist in identifying potential problems and risks in Agile meetings, and provide actionable and useful recommendations?

RQ2 To what extent do the AI meeting-assistants generate sensible recommendations in the context of real meetings in real time?

RQ3 How do users perceive the AI meeting-assistants in terms of user experience, and what impact does it have on overall performance?

Section 2 provides a brief overview of the relevant academic and grey literature on AI for software development tasks. Then, the paper moves on to detail the methodology used in this paper in Sect. 3. Section 4 analyses the interventions and evaluation surveys undertaken during this study, and reasons about the design decisions that lead to the final AI meeting-assistants. Finally, Sect. 5 points out the lessons learnt and how they could be transferred to similar contexts, and Sect. 6 highlights important implications for future practice.

2 Related Work

2.1 Adapting Agile Practices to Companies

Many organisations have already shifted from traditional working processes to Agile. Nevertheless, not all agile teams follow guidelines thoroughly [7,12]. This is because they might feel that following all Scrum rules is too time-consuming or that some of the Scrum rules are irrelevant or even outweighing the benefits [14].

Mortada et al. discuss several wrong practices in Scrum teams. For example, daily Scrum meeting is supposed to last only 15 min, but it often runs longer than that [12]. Other examples include not estimating user stories, not having the correct structure for writing the user stories in the backlog, not having a product backlog, not defining the sprint goal at the beginning of the sprint and not ending the sprint with demonstrating the desired deliverable [12].

Similar problems might also be present in Scaled Agile setups. Previous research has highlighted the role of confusion about roles and responsibilities in creating of unnecessary overhead [14]. For example, senior engineers and architects, might want to get more time to understand technical details, which requires separate smaller feature estimation meetings [14]. At the same time, developers often do not receive feedback from the right stakeholders at the end of the sprint [12].

For these reasons, some companies have proposed modifications and ad-hoc adaptations to the guidelines. For instance, Austrian Post proposes the Post-Rolling Refinement Model (PRIME) [14]. PRIME aims to remove a lot of the bureaucratic overhead, pushes decisions to the Scrum teams, and reduces the number of meetings and the number of mandatory participants [14].

2.2 Generative AI in Software Engineering

Generative AI (GenAI) is a type of AI that can generate different types of content, such as images, text, audio, and 3D models, based on the input it receives. Large Language Models (LLMs), such as *GPT-4*, are currently being used in a wide range of fields, including medicine, economics, software development, academia, and business. GenAI can also support software engineering [13]. Managers could use GenAI tools to get recommendations for decision-making, or to automate some interactions with the customers, e.g., using virtual assistants integrated with customer service tools etc [15]. Organisation's data could also be used as input to the AI tools for the purposes of creating tables, analysing statistics, generating models, and monitoring workflows [10].

Microsoft breaks the interventions of AI in before, during, and after team collaboration sessions and meetings and provides insights into how organisations can use AI to retrieve or generate relevant information and resources [10]. However, there are challenges associated with integrating AI and Agile methodologies, such as the need for specialised technical expertise, and integrating AI into Agile software development processes requires careful consideration of the context [6]. Moreover, the most critical challenges are related to human factors, e.g., ensuring that developers have the skills to work with AI and align expectations effectively [4,8]. As a result, the trade-off between creativity, human oversight, and cyber-security is a critical factor to consider.

2.3 Prompt Engineering

Prompt engineering is defined as a set of techniques to improve the inputs or instructions that a user provides for an AI model to get desired outputs [13]. Depending on the context that the AI model is used for, designing appropriate prompts is very important to get more accurate results, therefore it is suggested to narrow down the prompts and avoid using too general queries [5]. Mastropaolo et al. also studied the influence of varying natural language descriptions for Copilot prompts and proved that paraphrasing leads to different quality levels for the generations [9]. There are different techniques for prompt engineering including role-prompting, user of triple quotes to separate, trying several times with generating responses, etc [2].

It is known that it is difficult to balance relevant information retrieval (in this case, generating recommendations) and not overloading the participants [1]. These two things can be balanced in the prompt.

3 Research Design

The study aims to understand how practitioners use and perceive the use of AI in meetings, a relatively new phenomenon [10]. We adopted action research to observe both technical and social aspects of AI usage through interventions in *Austrian Post Group IT*'s online meetings. The emphasis of this study is not the artefacts produced (provided as Supplementary Materials), but rather the motivations behind design choices, detailed in Sect. 4. Before that, Sect. 3.1 provides some context, Sect. 3.2 describes the expectations for both assistants, and Sect. 3.3 explains how surveys and observations were used to draw conclusions and improve the artefact.

3.1 Context

Austrian Post is an international postal, logistics and service provider operating in the markets of Austria, where it plays a critical role in the country's infrastructure, eight other countries in Central and Eastern Europe, and Turkey. The development teams, which are part of *Group IT* use broadly accepted frameworks, such as Scrum as well as a in-house designed Scaled Agile framework named PRIME [14]. More than 520 employees work at *Austrian Post Group IT*, out of which approximately 300 have a similar role as the participants, that belong to 3 out of the 9 development teams at the *Digital Logistics Platform.*

An action team, consisting of both practitioners and researchers, was named to be responsible for planning, executing, and evaluating the research. The selected practitioners were involved in planning and executing actions, besides observing and providing contextualised feedback after each of the interventions. Moreover, they provided the evaluation of the final outcome with their deep knowledge of the context, the Scrum and PRIME frameworks, and the practitioners' way of working.

Complementary, a reference group of practitioners, responsible to give advice and feedback to the action team, and a management team, who is planned to govern the institutionalisation of the proposed changes, were also key to conduct the present study. The goals of these two groups is to evaluate the benefits of AI meeting assistants, as reported in this study, and exploring potential directions for further work, aligned with *Austrian Post*'s strategic goals, e.g., automating repetitive preparation tasks, supporting less experienced developers, creating useful summaries for those that could not attend a meeting, etc.

3.2 Action

We propose two AI assistants to use before, during and after two Agile software development meetings, as shown in Table 1. The assistants are instanced by prompting *Azure OpenAI Studio*'s *GPT-4* LLM. The prompts were designed iteratively: first listing the current challenges of each meeting, then refining the prompts and testing them without sharing the generations (silent demos), and finally using the AI assistants with participants, under observation.

The Agile Release Train Coach Assistant. An AI assistant was designed, with the help of reference team, to help the *Agile Release Train Coach* (a servant leader to the train and support teams in delivering value) prepare and conduct PRIME meetings by helping refine and plan the next PRIME iteration [14].

Table 1. AI meeting assistant actions

Intervention	Before meeting	During meeting	After meeting
PRIME meeting	Creation of slides with identified risks based on current PRIME features board		-
Daily Scrum	-	Real time guidance, ticket management	Post-meeting recommendations

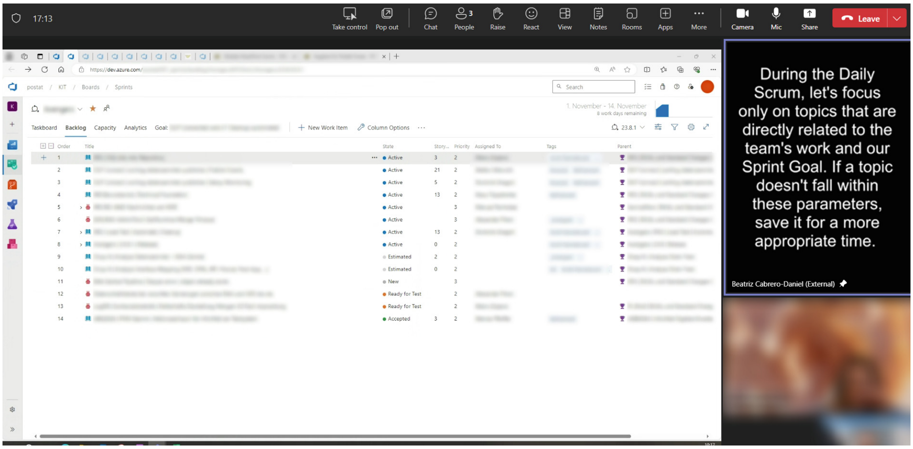
The *Agile Release Train Coach assistant* is instantiated using a prompt and three spreadsheet files. The files contain information about (i) the features and related children User Stories in the PRIME feature board, (ii) the average velocity of development teams per sprint, and (iii) of the Agile Release Train [14]. These files need to be provided to the assistant given that no real-time connection to *Azure DevOps* is yet available for the LLM. The files are automatically embedded by the *Azure AI search* platform in order to be accessed by the LLM [11]. Using the embedded files, the assistant provides valuable insights to:

1. Limit the risk of teams over-committing (using team's capacity and velocity).
2. Identify features with no effort value.
3. Identify features placed in incorrect backlog (based on iteration path).
4. Check unplanned integration testing efforts.
5. Identify features where there is a children user story for a team that is not tagged in the feature.
6. Help plan large features (based on effort points).
7. Highlight non-estimated and incorrectly estimated features.
8. Limit the risk of the Agile release Train over-committing (based on capacity).

The information contained in the files was manually gathered from *Azure DevOps* and anonymised by the reference team in two-hour-long sessions before each of the PRIME meetings described in Sect. 2. In these sessions, the prompt to instantiate the *Agile Release Train Coach assistant* was improved and the validity of its insights, checked. It is important to note that not only was the AI assistant faster than the reference team at analysing the data, but also made less mistakes than humans. Moreover, as the prompt design improved, as discussed in Sects. 4, the time to generate and check the insights went down to 30 min.

Scrum Team Assistant Tool for the Daily Scrum. Based on the insights of the action and reference team, a second assistant was designed to get real-time recommendations on the meeting progress, and insights on the adherence to the official Scrum guidelines right after the Daily Scrum.

The *Scrum Team Assistant Tool* assistant is instantiated using a prompt and the latest version of the official Scrum Guide, since there is a risk of the



During the Daily Scrum, let's focus only on topics that are directly related to the team's work and our Sprint Goal. If a topic doesn't fall within these parameters, save it for a more appropriate time.

Beatriz Cabrero-Daniel (External)

Fig. 1. AI recommendations, shown using *OBS*, during an Daily Scrum meeting.

LLM retrieving a deprecated version. The prompt gives the general context of the intervention and instructs the LLM on how to act depending on the user inputs. To generate manageable, to-the-point recommendations and align with the expectations of practitioners, the assistant is asked to generate up to 10 words that are "friendly." The prompt concludes with general instructions for the LLM to run the assistant. Once instantiated using the prompt in Supplementary Materials, the AI assistant generates different recommendations when:

- An individual talks about topics not related to the team's work.
- The work that the individual mentions is not visualised in the sprint backlog.
- An individual engaged in a detailed discussion about a specific topic.
- The impediment that an individual raised is not visualised.
- An individual was interrupted by an external circumstance.
- An individual does not have any task in "updated state" in the backlog.
- An individual needs to create a ticket to a specific team.
- Any other problem that may occur during the Daily Scrum.

The real-time generated recommendations are shared with the team using disappearing pop-up messages, as represented in Fig. 1. Right after the meeting, as shown in Table 1, the assistant provides a summary of the problems the team has faced during their Daily Scrum, lists all created tickets, and shares tips how to improve the next Daily Scrum Meeting via the meeting chat.

3.3 Data Collection and Analysis

The observations, surveys, and discussions held with practitioners helped capture the nuances of the practitioners' opinions and provide rich data for analysis.

Surveys to Understand Team Composition. Previous to the interventions, we sent a survey to the Scrum teams in the *Digital Logistics Platform* to capture the practitioners' opinions on AI. The participants' responses (a total of 39) helped us to select three teams willing to participate in the interventions. The three selected teams, as shown in Fig. 2, had a similar composition in terms of their feelings about AI meeting assistants. The readers who want to replicate the study can use the survey questions, provided as Supplementary Materials.

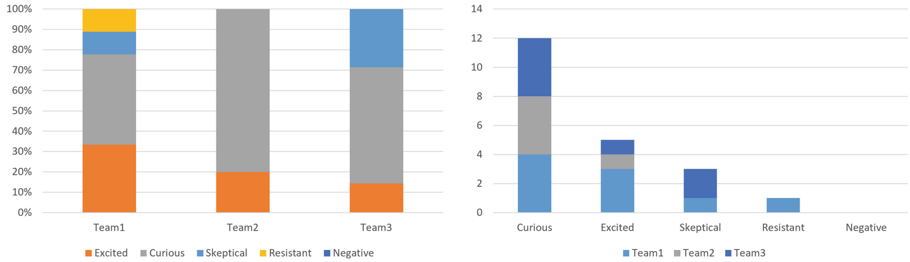


Fig. 2. Team composition, based on their answers to the initial survey question "How would you feel about using AI support in the meeting?"

Observations Before and During the Actions. The researchers in the action team participated on the online meetings over a period of time to get accustomed and to understand their routines and needs [16]. During the observations of the meetings, the researchers did not intervene, even though the meeting attendants were aware of their presence. In each of the interventions and silent demos for the two assistants, notes were taken by the action team about the AI-practitioner interaction. Being part of the environment also helped interpret the true meaning of the answers obtained before and after each of the interventions [16].

How the Conclusions Were Drawn. The survey answers, together with the feedback by the practitioners after the interventions, and the notes from the observations were used to improve the design of the prompts, as discussed in Sect. 4.

4 Execution and Results

This section first presents the specific challenges identified during the initial observations, previous to the interventions, and suggests how AI could assist practitioners (**RQ1**). The participants views, collected after each intervention, helped improve the AI assistants in helping reach the meeting goals and adhere to the guidelines (**RQ2**). Finally, this section moves on to discuss how the practitioners felt about the AI assistants and the implications on their way of working (**RQ3**). Themes emerging from the participants' responses are also highlighted here and their transferability to other contexts is discussed in Sect. 5.

4.1 Preparations with the Agile Release Train Coach Assistant

At the beginning of each PRIME iteration, information about the features for the next PRIME needs to be prepared to allow for discussions during the meeting. Doing this is, according to one of the practitioners, a "boring process" that takes considerable time from a number of people. As a result, and similarly to the three groups represented in Fig. 2, most of the participants on PRIME interventions (76%) also reported feeling excited or curious about testing AI assistants.

To help with preparations, the *Agile Release Train Coach assistant* provides insights meant to help the work, as reported in Sect. 3.2. These insights are described to the LLM in the prompt, provided as Supplementary Material, that was iteratively designed in the three action iterations in Table 2.

Table 2. Record of all notable changes made to the *Agile Release Train Coach assistant*

Intervention	Feedback/reasoning	Changes
Initial tests	-	Prompts for benefits 1 to 6 based *Azure DevOps* data
PRIME 1	Metrics confusing and do not capture feature dependencies, observation: difficult to share	Fixed benefits 7 and 8, changed prompt to generated slides to use in the discussion
PRIME 2	Wrong velocity and capacity calculations (character mismatch)	Improved definitions for key terms, fixed team name mapping
PRIME 3	Data is outdated (hours)	-

In the **first intervention**, the pre-generated insights were tabulated and shared with the practitioners, who required many clarifications on the values in the tables (e.g., how were effort points used to calculate the average team velocity), but overall agreed with the usefulness of the *Agile Release Train Coach assistant*. In this first try, however, a third of the generations were mistaken due to an outdated input data point due to a human mistake. When the errors were spotted, a participant reminded their team that "these things were done manually before" and called the AI "very handy" even if it makes minor mistakes.

Right after the meeting, participants were given the chance to provide feedback through an anonymous online survey and reported some potential misinterpretations of the AI generations. For instance, one participant stated that "a team might over-estimate but the train might still be under the [effort] threshold" and another requested clarifying the relationship between "teams' capacity, Train velocity, and user story point estimation." Their feedback, together with comments and notes taken during the intervention, lead to changes in the prompt. First, "risks" were renamed to "benefits" to align with the goal: helping practitioners by informing their discussions, rather than unilaterally sending warnings. Moreover, the mathematical operations were rephrased and clarified,

Table 3. Record of all notable changes made to the *Scrum Team Assistant Tool*

Intervention	Feedback/reasoning	Changes
Silent demos	-	Requesting positive messages, "no problem" option added
Team 1	Aggressive, not synced, and distracting	Shorter messages, rephrased "warning", asked for friendlier generations, pop-up messages
Team 2	Happy with non-intrusive messages	Break prompt down, interface improvements
Team 3	Recommendations partially incorrect, easily "ignorable"	-
Team 1, validation	Real time messages and suggestions, helpful; summary, appreciated	-

and two benefits were added following the recommendations from the *Lead Product Manager*, responsible for the products of the train and owner of the train backlog [14], on how to identify features that have unrealistic estimation. Finally, the prompt was modified to generate slide templates.

During the **second intervention**, the slides created with the help of the *Agile Release Train Coach assistant* were used to present the insights described in Sect. 3.2. Again, some practitioners questioned the calculations and logic behind them, e.g., whether "only features with estimated efforts are used." Another practitioner, after hearing the details about how potential over-commitment is computed by the assistant, asked: "does it mean that we have to change our way of working?" These comments led us to further refine the definitions within the prompt in order to have clearer AI generations for the questioned benefits.

Some problems arose during the second intervention regarding the mathematical computations due to an inconsistency in the anonymisation processes: there was a character mismatch in the teams' names between the spreadsheets that are used alongside the prompt. As discussed in Sect. 5, the used LLM has troubles with mathematical operations, and therefore they need to be defined in a very precise way. Due to this, further reformulation of the benefits was done to reach the final, unambiguous version of the prompt.

In the **third and last intervention**, all insights were well received and a single negative comment arose: the data was a couple of hours old. Given the LLM's performance limitations and the lack of connection to *Azure DevOps* data, the data needed to be prepared the morning before the intervention. Throughout all the interventions, participants repeatedly pointed out the potential of "looking at [the recommendations] live." However, having the *Agile Release Train Coach assistant* working with real-time data is outside of the scope of this study and left for future work, as discussed in Sect. 5.

4.2 Real-Time Assistance by the Scrum Team Assistant Tool

We also propose, as presented in Table 1, AI assistance during and after Daily Scrum meetings. To prepare the interventions, the action team observed the practitioners in their online meetings prior to the action taking. These observations, together with the insights by the reference team, helped design the recommendations listed in Sect. 3.2. With this information, the *Scrum Team Assistant Tool* was designed to help participants follow the official Scrum guidelines.

In order to refine the prompt used to set up the *Scrum Team Assistant Tool*, three teams help us perform four design iterations, as reflected in Table 3. The composition of the three teams is similar to Team 1 when it comes to pre-conceptions about AI, as can be see in Fig. 2. Even though team members reported differences in their preferred way of working, we treat the feedback received from each team as applicable to others.

In the **first intervention**, the AI recommendations were shared via the *Microsoft Team*'s chat and the members of *Team 1* found the amount of messages overwhelming and "rather distracting." The first message caught the participants' attention, however, they did not seem to mind the subsequent warnings: one participant even stated they were just "random messages," and another complained that they were "warnings, warnings, and more warnings!"

In the survey, sent after the meeting, participants reported that the generations were aggressive and "missing empathy," and 2 out of the 7 participants reported feeling annoyed by the AI. Even so, more than half of the respondents (4 out of 7) reported liking being warned when the team engaged in too detailed discussions and being notified when the *Scrum Team Assistant Tool* estimated that the Daily Scrum would take longer than 15 min. It is important to note that, in the survey previous to the interventions, only 24% of participants said their team does not usually finish their Daily Scrum within the 15-minute timebox. This contradicts the findings by Mortada et al., that reported 53% of Daily Scrum events not finishing within 15 min [12].

Using the feedback, the prompt was changed to generate shorter messages and not to produce "warnings" but "recommendations," which proved to change the tone of the generations (i.e., friendlier, tactful). After the first intervention, we also improved the interface to make it less distracting: we introduced disappearing pop-up messages, as seen in Fig. 1. The new prompt and interface for the *Scrum Team Assistant Tool* were tested with *Teams 2* and *3* in the subsequent interventions, and were well received. After seeing the changes, *Team 1* was also more willing to have an AI assistant than after the first intervention.

Before the **second intervention**, the prompt was extended to generate a summary of the recommendations at the end. All prompt improvements were tested with *Team 2* and the participants overall liked the experience and said the AI recommendations were non-intrusive. No problems were observed during the meeting or reported afterwards, however one of the practitioners reported minimising the view with pop-up messages (so they were not readable).

The **third intervention**, with *Team 3*, was similarly successful except for two specific issues. On the one hand, a participant stated that some of the generated recommendations about "keeping the discussion on the topic felt partially incorrect." This was interpreted, with the help of the reference team, as a team-level preference; while the LLM makes recommendations to strictly follow the official Scrum guidelines, different teams had preferences as to what to allow in their Daily Scrum meetings (e.g., *Team 1* accepts social related talk, as long as it stays within 15 min). On the other hand, a participant suggested that the "Scrum Master should keep an eye on these things" and have a final say on what AI recommendations are shared with the team.

In the **last intervention**, the participants agreed that the *Scrum Team Assistant Tool* "provided helpful live messages, both positive and negative." However, similar to a participant in *Team 3* that reported "feeling observed during the meeting," one of the participants explained that "it feels unnatural" to have "something inhuman forcing [...] a specific pattern on us." In general, across interventions, the participants appreciated the summary of the recommendations received at the end, and participants stated they "would like to have this summary for all other meetings" and suggested it would be helpful to expand it with "what was done well and if it has improved since last time." These comments and their implications for future work are further discussed in Sect. 5.

5 Discussion of the Implications

Several reports have shown the potential of using AI to enhance various Software Engineering tasks. As mentioned in Sect. 2, prior studies that have noted the importance of appropriate interfaces and human oversight when integrating AI in Software Engineering [3]. However, very little was found in the literature on the question of how to design AI assistants for meetings and integrate them in real industrial setups. The lessons learnt in this study are discussed below, and how they could be transferred to other contexts is discussed in Sect. 5.1.

RQ1 sought to determine how to create AI assistants to help identify and address potential problems and risks in agile meetings, and suggest improvements. By observing different teams during Daily Scrums and PRIME meetings [14], challenging areas where AI could assist human practitioners were identified. Then, different interventions, in Tables 2 and 3, were used to design the *Agile Release Train Coach assistant* and the *Scrum Team Assistant Tool*.

Take-away message: AI should not warn, but inform about potential improvements and give helpful suggestions, so they are not negatively perceived.

RQ2 focused on the design process leading to sensible AI recommendations to use before, during and after the meetings. Therefore, a number of design iterations, described in Sect. 4, were used to determine the effect of different prompting strategies in the performance of the LLM in doing so. Overall, both assistants are able to generate accurate and contextualised insights, surpassing the expectations of some of the participants.

Take-away message: AI assistants can be useful even if mistakes are made, and practitioners are to expertly review the generations. AI should work alongside practitioners, not replace them.

RQ3 focused on the social aspect of AI meeting assistants. In the first iterations, as described in Sect. 4, the proposed action was not well received and multiple iterations were needed to design balanced solutions. The results are in agreement with recent studies that highlight the important of appropriate interaction strategies in promoting trust in AI tools [3,4]. It is interesting to note, though, how the participants' perception of AI depended on how seamlessly it was integrated in the meetings, and its insistence and intrusiveness negatively impacted the participants' perception of them: from useful to imposing.

Take-away message: Each practitioner and team feels differently about AI recommendations on adherence to Agile principles and practices. Therefore, AI meeting assistants should be adapted to their needs and expectations.

5.1 Recommendations for Company and Teams: Readiness Assessment

The emphasis of this study is not on the artefacts produced but rather on the procedural aspects of utilising these tools and the transfer of the lessons learnt to other industrial settings. This section moves on to present a set of actionable recommendations for other companies for how to apply the results presented here, and how to conduct similar studies in the future. These insights are gathered in the *Readiness Assessment of Human-AI Collaboration for Agile Meetings* form, provided as Supplementary Materials, to guide interested companies and teams.

Customised Team-AI Interactions: The findings of this study highlight the need to adapt the AI assistants to each team, which needs to be studied prior to considering integration. Data about the teams' needs, expectations, and preferred Agile practices should be gathered to customise the AI assistants. This technology, however, should be imposed neither on teams nor on team members. Therefore, the practitioners' feelings on AI must also be assessed beforehand: if there is opposition, receiving LLM-specific training might help. Still, after integration, feedback should keep being gathered, as presented in this study.

Teams looking to integrate AI assistants to enhance Agile team collaboration sessions, should assess their readiness using these questions:

□ Have you assessed what your team's challenges are regarding agile meetings?
□ Have you evaluated the team-specific agile practices and adoption maturity?
□ Have you gathered data about practitioners's feelings on AI-assistants?
□ Does the team have the knowledge to integrate AI assistants(s) in meetings?
□ Does your team see any benefits if integrating AI assistant(s) into meetings?
□ Can the team to provide iterative feedback to improve the AI-assistant?

AI Assistant Design: During the design phase, a difficult question arose: whether AI assistants should be a support for everyone in the team or for a specific role, only. Companies (or teams) looking to integrate AI assistants need to reflect on what their goals are and design them appropriately. The authors' recommendation is to not attempt to cover all possible functionalities; but rather use different modules, or *agents*, that connect only when needed. Moreover, each of the modules should provide input and pointers to the practitioners without contradicting Agile values. Once integrated, the AI assistant's design should be rethought and improved based on the periodically-received feedback. Companies (or teams) should assess their early AI assistant design using these questions:

□ Does the design of the AI-assistant contradict with any agile principles?
□ Is the AI-assistant designed to support the Agile team or just a specific role?
□ Is the AI-assistant designed to enhance or substitute a practitioner's role?
□ Is the AI-assistant design modular and scalable?

Investments and Compliance: Before AI assistant integration, companies should consider whether the required expertise is available and whether the investment can be made. This includes economical resources but also reflecting on the long-term impact of using these technology (e.g., on societal and environmental well-being [3]). Then, data protection and security concerns must be addressed (e.g., GDPR, user permissions, etc.). For this, as discussed in Sect. 4, we recommend ensuring that the LLM has secure access to up-to-date data (e.g., deprecated versions of online documents were often retrieved during our tests). Companies should therefore assess their readiness by asking:

□ Does the company have the expertise to integrate AI-assistants?
□ Is the company willing to invest resources into creating AI-assistants?
□ Is the integration of the AI-assistants compliant with the company's code of conduct, and ethics and sustainability strategies?
□ Does the company have available interfaces to connect AI-assistants to other internal systems, and guarantee the retrieval of up-to-date data?
□ Can AI-assistants be configured to only use reliable external data sources?
□ Is the integration of AI-assistants compliant with the company's AI-strategy, security standards, and privacy policies?

5.2 Threats to Validity

We must remind the reader that the claims are based on the actions conducted with the help of willing participants. The selection of the subjects, based on their *willingness to explore AI as an assistant* in meetings, might have introduced a bias. Moreover, evaluation apprehension may have caused these subjects to *behave differently when observed*, and feel more inclined to positively evaluate the AI assistants and pay more attention to the adherence to the Scrum framework under observation. Moreover, the meetings were conducted in English to accommodate for the action team. However, the practitioners often showed *more fluency when discussing issues in their native language*. This might caused delays or affected the normal conduction of the meetings.

The generalisability of this work might also be affected by the *diverse data protection policies* and available resources at each industrial setup. For instance, this study focuses on Azure OpenAI Services: *alternatives have not been explored* because of privacy concerns and case company constraints. Moreover, the provided prompts might not directly generalise to other setups for different reasons. However, the emphasis of this study is not on the prompts themselves, but rather the motivations behind design choices, and the process to gather feedback after each iteration. It is also worth noting that some events (e.g., popularisation of Microsoft Copilot) during the time that this study was conducted might have affected both the practitioners' and the management team's perception of the proposed AI-meeting assistants. Similar and unforeseeable events, might continue to shape their opinion on these promising tools.

6 Conclusion

The main goal of this study was to determine whether AI can assist practitioners in Agile meetings by identifying potential problems and risks. Moreover, the usefulness of the generated insights, and how they were perceived by the participants in our action research interventions, was also studied.

While the results of the paper, discussed in Sect. 4, demonstrate the usefulness of AI assistants in generating useful recommendations in real meetings, they also highlight the need to pay close attention to the user experience of practitioners that directly interact with AI. Although the current study is based on a relatively limited sample of participants, all from *Austrian Post*, this work offers valuable insights into the expectations of participants on AI assistance and on the strategies to integrate AI assistants before, during, and after meetings.

This study provides the first comprehensive assessment of how AI meeting assistants can be integrated in a real Agile setting, taking into account the perceptions of the practitioners in the design process. The findings, the authors hope, will be useful to guide the integration of AI into different team collaboration sessions and meetings in different setups.

6.1 Future Work

Further research might explore the adaptation of the AI assistants to specific needs of the teams they will assist. For instance, teams might have preferences when it comes to adherence to official Scrum guidelines or be at different stages of Agile adoption, in which case AI could help Scrum Masters specifically. AI could also suggest ad-hoc best practices or solutions for specific issues within a team. Moreover, the multi-modal capabilities of newer models in order to visualise information and inform practitioners more efficiently could be explored in the future, as practitioners seem to prefer graphical representations.

Similarly, further studies can be carried out to understand how AI can be integrated into other meetings, standard or ad-hoc, in other industrial setups. For instance, the reference team believes that practitioners would benefit from AI assistants in longer, resource intensive meetings, planning meetings, or review meetings, where the official guidelines might not suit some teams well. Moreover, AI could provide overviews across meetings and teams to help practitioners have an overview of the team, train, and whole development progress.

Future work could also focus, as suggested by multiple participants in this study, on generating meeting summaries for different purposes, such as informing missing participants about what transpired in a meeting. However, in order to provide relevant summaries and recommendations, the LLMs would need context such as automatic meeting transcripts and access to *Azure DevOps*. This data would make the recommendations more relevant and not outdated; however, further development, outside the scope of this paper, is required.

Finally, we believe a thorough assessment checklist should be crafted for companies and teams to understand their readiness to integrate AI-assistants in their Agile meetings. Then, and only then, should human-AI collaboration start.

Acknowledgements. Thanks to Prof. Staron for his time and his book. Thanks to Prof. Berger for his valuable guidance. Thanks to those that told us their honest opinion about AI.

References

1. Asthana, S., Sajnani, H., Voyloshnikova, E., Acharya, B., Herzig, K.: A case study of developer bots: motivations, perceptions, and challenges. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2023, pp. 1268–1280. Association for Computing Machinery, New York (2023)
2. Chen, B., Zhang, Z., Langrené, N., Zhu, S.: Unleashing the potential of prompt engineering in large language models: a comprehensive review. arXiv preprint arXiv:2310.14735 (2023)
3. Commission, E., Directorate-General for Communications Networks, C., Technology: The Assessment List for Trustworthy Artificial Intelligence (ALTAI) for self assessment. Publications Office (2020)
4. European Commission, Directorate-General for Communications Networks, Content and Technology: EUR-Lex - 52021PC0206 - EN - EUR-Lex. CNECT (2021)
5. Hörnemalm, A.: Chatgpt as a software development tool: The future of development (2023)
6. Karac, I., Turhan, B.: What do we (really) know about test-driven development? IEEE Softw. **35**(4), 81–85 (2018)
7. Kuhrmann, M., et al.: What makes agile software development agile? IEEE Trans. Softw. Eng. **48**(9), 3523–3539 (2022)
8. Larios-Vargas, E., et al.: DASP: a framework for driving the adoption of software security practices (2022)
9. Mastropaolo, A., et al.: On the robustness of code generation techniques: an empirical study on github copilot (2023)
10. Microsoft: The future of meetings: Using ai to improve team collaboration (2019) Accessed on November 30, 2023
11. Microsoft: Vector search in Azure AI Search (2023). Accessed 20 Dec 2023
12. Mortada, M., Ayas, H.M., Hebig, R.: Why do software teams deviate from scrum? reasons and implications. In: Proceedings of the International Conference on Software and System Processes, ICSSP 2020, 71–80. Association for Computing Machinery, New York (2020)

13. Nguyen-Duc, A., et al.: Generative artificial intelligence for software engineering–a research agenda. arXiv preprint arXiv:2310.18648 (2023)
14. Niessl, M., Gruber, C., Eder, M.: Restating scaled agile development at austrian post. XP (2023)
15. Sarker, I.H.: Ai-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems. SN Comput. Sci. **3**(2), 158 (2022)
16. Staron, M.: In: Action Research as Research Methodology in Software Engineering, pp. 15–36, January 2020

The Role of Team Composition in Agile Software Development Education: A Gendered Perspective

Gyda Elisa Sæter, Viktoria Stray(✉), Steffen Almås, and Yngve Lindsjørn

University of Oslo, 0373 Oslo, Norway
{gydaes,stray,steffa,ynglin}@ifi.uio.no

Abstract. Team composition is a critical factor influencing collaboration within agile software development. This study investigates the impact of gender distribution on teamwork quality in software engineering capstone courses. We examined the experiences of 240 students organized into 40 teams during an agile project course. We analyzed two surveys, one conducted before team composition and one at the end of the project work. As much as 91% of the students chose to use the practice of conducting stand-up meetings in their project work and the majority were satisfied with the practice. Further, our analysis reveals that while women tend to engage more in design and men in programming, an increase in the proportion of women within a team correlates with a higher involvement of women in programming tasks. Our findings highlight gender differences in perceptions and experiences related to project involvement in agile software engineering courses.

Keywords: Collaboration · Teamwork quality · Knowledge sharing · Human and social aspects of agile software development · Gender diversity · Team assembly strategies · Empirical study on gender effects · Teaching experiences · Empirical studies with students

1 Introduction

Software development is extremely collaborative, especially in agile development [3,10]. One of the important objectives of software engineering education is, therefore, to make students learn essential collaboration skills [20]. Teamwork in software engineering projects among higher education students is harder than anticipated [2]. Common challenges are a lack of dedication and involvement of one or more team members [17] coupled with communication challenges among the team members [11].

Designing courses that effectively teach agile software development and equip students with the skills to tackle real-world challenges remains a formidable task [8]. Team composition in large capstone courses presents instructors with a complex and time-consuming challenge, as they must balance a range of potentially

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, pp. 179–195, 2024.
https://doi.org/10.1007/978-3-031-61154-4_12

conflicting factors, including the distribution of skills and availability among students, and the varying degrees of enthusiasm and ambition for the project. However, creating well-functioning agile development teams is crucial for fostering learning, encouraging student engagement, and achieving a successful project outcome [21]. It is essential to consider team composition, including gender diversity, in capstone software development projects to enhance learning and project outcomes. Different kinds of diversity can make teams underperform because the team members may not understand or talk to each other well, which can cause disagreement or dissatisfaction [14]. Recent research has found that when team members feel safe, they are more inclined to help each other and exchange their knowledge [15]. Lack of trust is one of the main barriers for autonomous agile teams [31].

Research on team composition in agile software engineering capstone courses has highlighted the challenges faced by instructors in creating well-balanced and effective teams [11,27]. To address this, algorithmically supported systems like TEASE have been developed to assist instructors in team composition, resulting in better mean project priority and reduced time for team formation [11]. The importance of motivation, interpersonal relationships, and communication in team satisfaction has also been emphasized [11]. Furthermore, the use of Scrum in capstone courses has been found to positively impact students' estimation and planning skills [28]. Finally, the effectiveness of the proposed team building criteria in enhancing team cohesion has been demonstrated [36].

In this paper, we describe our methodology for assembling development teams, which is based on a predetermined set of criteria derived from our extensive experience in conducting the course since 2019. Many developers have their first encounter with agile software development through courses conducting teamwork during their education, where this teamwork experience sets the tone for developers' view of agile collaboration. Due to the low percentage of women in the sector, female developers frequently find themselves in the minority within their teams. In teamwork, social research found that minority team members encounter gender stereotypes, affecting their task assignment and performance [1,18,40]. Agile software development teams where women are the majority are currently understudied, leaving a knowledge gap regarding how different gender compositions affect teamwork. Amidst the extensive body of literature on gender, agile student teamwork, and team composition separately, there's a lack of research exploring the affection of gender composition in student teams in software engineering courses. Motivated by this gap, this study aims to explore gender influence on the quality of teamwork by answering the following research question:

RQ1: *"How can diverse student teams be composed in an agile software engineering capstone course, considering students' preferences?"*

RQ2: *"What gender differences do we find in an agile software engineering capstone course?"*

RQ3: *"How does gender composition affect the student's teamwork in agile student teams?"*

Our case study was designed to explore gender dynamics in an agile software development capstone course. The course involved 40 teams, and data was collected through two surveys. Our study is based on Kanter's theory [19] of gender diversity and the concept of teamwork quality (TWQ) as defined by Hoegl and Gemuenden [16], focusing on the quality of interactions within the team. Our measure of teamwork quality does not encompass the assessment of task process, task strategy, or the individual team members' performance quality in carrying out task activities. Additionally, management-related activities like task planning, resource allocation, and management by objectives are not within the scope of this teamwork quality construct.

In the following sections, we will present the theoretical background used in this paper in Sect. 2 and the context and methodology of the case study in Sect. 3. Section 4 presents our results, structured by the research questions, and Sect. 5 discusses the results before concluding in Sect. 6.

2 Background

2.1 Diversity in Software Development Teams

Early social research stated that women prefer working in either gender-balanced or male-dominated environments over women-dominated environments [41]. However, recent studies in software engineering propose that male-dominated environments, and lack of female support, drive women away from the field [13,29]. A recent estimate of the percentage of women among the world's software developers is approximately 10% [7]. The underrepresentation of women in certain academic disciplines is accompanied by the risk of female students disengaging or dropping out, posing a significant challenge to educational institutions [13,29].

Teamwork is one of the most critical soft skills in software engineering, often taught through project work in capstone courses where students collaborate in teams [4]. Curseu et al. [9] recommend forming diverse teams based on gender and nationality to enhance learning performance, as diverse groups tend to perform better. However, a study by Aeby et al. [1] involving third-year Bachelor and first-year Master students found significant gender differences in role preferences: females were more likely to engage in report writing, while males showed a stronger inclination towards technical tasks.

Løvold et al. [27] found that student-formed teams performed slightly better than instructor-formed, advantaging from already knowing each other and how they work together. However, mixing the two approaches, allowing students to submit their team preferences for the instructor to base team formation on, could give the students ownership and include social benefits from the self-assigned approach, while at the same time ensuring balanced teams regarding the project needs [33].

2.2 Teamwork Quality

As described in the Introduction, our use of the term Teamwork Quality (TWQ) is based on Hoegl and Gemuenden [16], and focuses solely on the quality of interactions within the team. The six subconstructs of communication, coordination, balance of member contribution, mutual support, effort, and cohesion cover performance-relevant measures of internal interaction in teams. Hoegl and Gemuenden used the TWQ instrument in studies of traditional software teams. In recent years, TWQ has been used in studies of agile teams [25,26]. See Appendix B for the items used in the TWQ construct.

2.3 Kanter's Tokenism Theory

In 1977, Kanter introduced the theory of Group proportions, also known as the Tokenism theory [19]. This theory argues that the dynamics of teamwork will be affected by the numerical proportion between the minority and the majority. Meaning that women in a minority position will be more exposed to stereotypes proportionate to the number of men in the team. The theory describes four types of gender compositions: uniform, balanced, tilted, and skewed. Specifically, *Uniform* are 100% homogenous gendered groups, *Balanced* groups have a ratio between 60:40 and 50:50, *tilted* groups of 65:35, and *skewed* groups have an 85:15 ratio. Within skewed groups, the minority is defined as "tokens" and the majority "dominants". They are called tokens as they are often made representative of their genders, following expectations and stereotypes from their gender into the person.

The increased visibility can result in tokens experiencing performance pressure and constraints in social behavior [23]. This might lead to token women assimilating the men dominants [22]. In tilted groups, women can form alliances and therefore influence the group's culture, while balanced groups are more prone to group dynamics depending on the individuality of the group members.

3 Methodology

3.1 Context

We aimed to investigate team dynamics with a gender perspective through a single-case holistic study [35,42]. The data was collected from January to June 2023 through two surveys distributed in an agile capstone course. The course is a compulsory fourth-semester course that provides 20 ECTs (equivalent to one-third of a full-time academic year's workload). The course is mandatory for three study programs at the Department of Informatics at the University of Oslo. Regardless of study program, all students are expected to participate in all aspects of agile software development.

The students are divided into teams tasked with developing applications based on the Norwegian Meteorological Institute's weather programming interface over a twelve-week period. Figure 1 illustrates the course's timeline for 2023. In that year, there were 247 students enrolled in the course, and 34% of students were female. These students were organized into 40 teams.

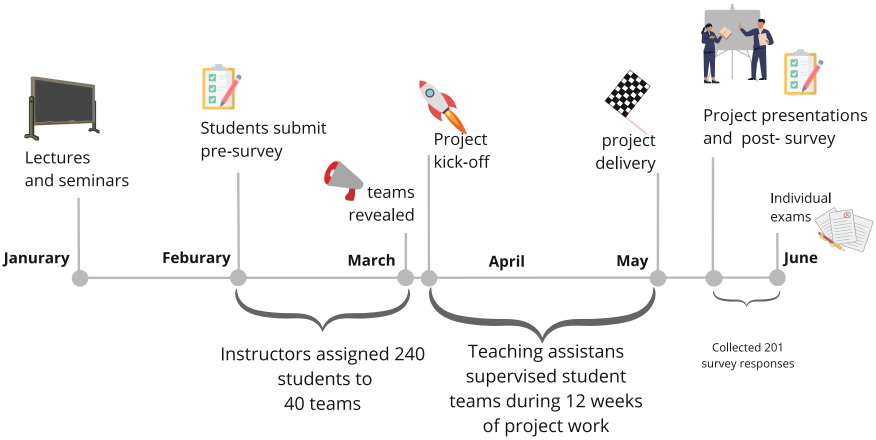


Fig. 1. Timeline Showing Course Structure and Data Collection

3.2 Data Collection and Analysis

Pre-survey: The team assembly was based on a pre-survey (Appendix A), requiring students to sign up for the teamwork, either alone or with a pre-assembled team of up to 3 students. The survey was only filled out by one student per pre-assembled team, requiring prior agreement among team members regarding project ambition. The survey was handed out in February with a 10-day submission deadline. As gender was not collected in the survey, members' gender was assigned by manually checking their registered gender in the student system.

Post-survey: In June 2023, a quantitative subjective survey was conducted to measure 1) the team members' self-assed teamwork quality, success, and performance, and 2) students' self-assessed perception of gender biases and gender compositions' influence on teamwork. The survey consisted of 82 questions, where 61 of the questions measured teamwork quality. Seven Likert scale questions asked about their self-assessed experience with gender bias within their teams during the project work, and 1 open-ended qualitative question inviting students to reflect on their teams' gender compositions, their effect on their project, and experiences with gender dynamics in teamwork in general. The full survey is available in Appendix B. The survey was answered by 201 respondents (response rate was 84%).

The survey was handed out to the students in between team project presentations, providing them with an environment exclusively related to their work regarding location, time, and headspace. The students were obligated to present at a given time, and dedicated time was allocated for conducting the survey, hopefully motivating them to answer it more thoroughly.

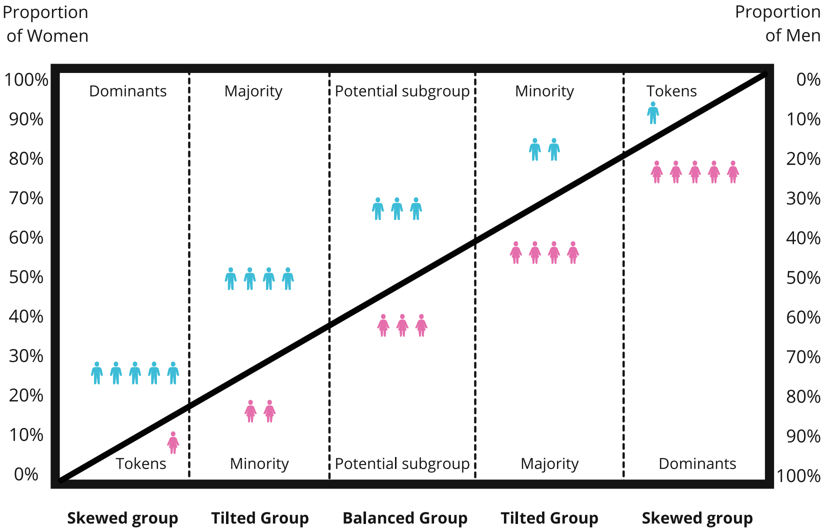


Fig. 2. Our implementation of Kanter's Tokenism Theory

Students are asked to select their gender provided with the following options: *Woman*, *Man*, *Other*, or *Do not wish to declare*. We are interested in the student's self-defined gender and have therefore employed the terms *Man* and *Woman* throughout the research as we believe them to be more inclusive than *female* and *male*, inviting students to identify regardless of biology [24]. Only one student chose *Do not wish to declare*, and was removed from the analysis due to data-relevance. Figure 2 displays this case study's implementation of Kanter's theory.

When analyzing the survey, the data was prepared for three main analyses: a) Teamwork quality aggregated by teams, b) Individuals' experience with teamwork and c) qualitative teamwork experiences. Sets a) and b) represented quantitative data, analyzed with statistics in SPSS and Python. Set c) was analyzed using reflective thematic analyses [6]. Thematic analysis is a systematic process for identifying, organizing, and gaining insights into recurrent themes within a dataset. By pinpointing common discussions or written content, shared themes and patterns are fabricated through this method.

4 Results

Of the initial 247 students who signed up for the course, 240 students qualified for the project work after passing all individual compulsory assignments, making 40 teams each with 6 members. Team assembly took the first author over 60 working hours, manually assembling by following variables in order of priority: 1) requested peer students, 2) level of ambition, 4) abnormalities in working hours 3) multidisciplinary, and 4) gender. This section presents the result of the team composition regarding these variables.

The pre-survey received 134 responses, where 34 pre-assembled teams with three members and 35 teams with 2 members. One team was permitted to pre-assemble with 6 members, due to challenges in working hours. Team assembly began with prioritizing a balanced composition regarding the number of members in the pre-assembled teams. Meaning, ideally, a team would either compose two pre-assembled teams of three peers, four of two peers, or six single students.

Pre-assembled teams usually compose friends, and we were concerned this would lead to imbalanced team dynamics, especially when three strangers join three close friends. Despite these concerns, we managed to form 23 symmetric teams where the composition mirrored our balanced ideal and 17 asymmetric teams where such balance could not be achieved.

4.1 Team Compositions and Ambition Levels Based on Pre-survey

In our survey, we inquired about the students' desired time commitment to the course, referred to as their ambition level (detailed in Appendix A). None of the surveyed students reported having ambition level 1 ("Wish to work much less than expected (<15 h/week)"). When comparing the reported ambition level among genders, we found that women reported a higher average ambition level, with a mean score of 3.91 (SD = 0.757). Men, on the other hand, demonstrated a slightly lower mean ambition level of 3.68 (SD = 0.778). The standard deviation among women's scores indicates slightly more variation in comparison to men's scores. These results suggest that, within this survey's population, women have expressed a marginally higher ambition level than men.

We strived to compose teams with team members having approximately the same ambition level. Teams were distributed as follows: two teams at level two, 12 at level three, 20 at level four, and six at the maximum ambition level; level 5 ("Wish to work much more than expected (>35 h/week)"). Table 1 illustrates the distribution of teams based on their ambition levels and gender composition, with each column representing the gender compositions (W=women, M=men).

Table 1. Distribution of teams by ambition level and gender composition.

Ambition	0W/6M	1W/5M	2W/4M	3W/3M	4W/2M	5W/1M
2	1	1	0	0	0	0
3	2	3	3	1	3	0
4	4	7	1	5	2	1
5	0	1	1	1	2	1
Total	7	12	5	7	7	2

Gender distribution was the last consideration in the team assembly, only adjusted if feasible after accounting for ambition levels, work hours, and study program. Consequently, only two teams comprise token men and no team comprises women uniformly. Given the overrepresentation of men in the course, forming all-women teams solely for the research would disproportionately increase the number of uniform men teams. Table 1 gives an overview of the number of teams in each gender composition.

4.2 Agile Ways of Working and Gender

In the post-survey, as much as 91% reported that they used the practice of conducting stand-up meetings. With regard to satisfaction with stand-up meetings (rated on a scale from 1, very dissatisfied, to 5, very satisfied), only 4% said they were "very dissatisfied," and 5% said they were "slightly dissatisfied." 11.9% were neither happy nor unhappy. In other words, the majority appeared to appreciate the stand-up meetings; 35.3% said they were "slightly satisfied," and 34.8% said they were "very satisfied" with the practice, while 9% did not utilize stand-up meetings. Women reported higher satisfaction with stand-up meetings than men, see Table 2. As can also be seen in the table, women reported spending more hours on the project work. The students also rated the degree to whether they had been a Scrum Master/Team leader (on a scale from 1 - Has not been to 5 - To a large extent), and women reported to have been Scrum Masters to a greater extent. However, these differences are not statistically significant.

Regarding the use of collaboration and communication tools, 46% used Discord, 52 % used Facebook Messenger, 40% used Microsoft Teams, and 40% used Slack. Almost 70 % used Trello, which might be explained by the high usage of ScrumBan/Kanban. Among the other responses, the most commonly mentioned collaboration tools were GitHub, Figma, Google Drive, Confluence, Snapchat, ClickUp, Signal, Jira, Overleaf, and Notion.

Table 2. Stand-up meetings and Scrum Master analyzed by gender

	Women		Men	
Metric	Mean	SD	Mean	SD
Hours Spent on Project	20.48	7.87	19.69	9.26
Satisfaction with Stand-up Meetings	4.13	1,13	3.95	1.05
Degree of being Scrum Master/Team Leader	2.95	1.22	2.67	1.02

4.3 Gender Differences in Primary Work Functions of Team Members

In the post-survey, students were asked what their primary work function in the team was, selecting either *programming*, *testing*, *designing*, *documentation*, *architecture*, or *others*. The survey received 201 responses, where 98 students reported programming as their main function, 45 reported designing, 30 reported documentation, 9 testing, 11 architecture, and 8 chose others. When we split

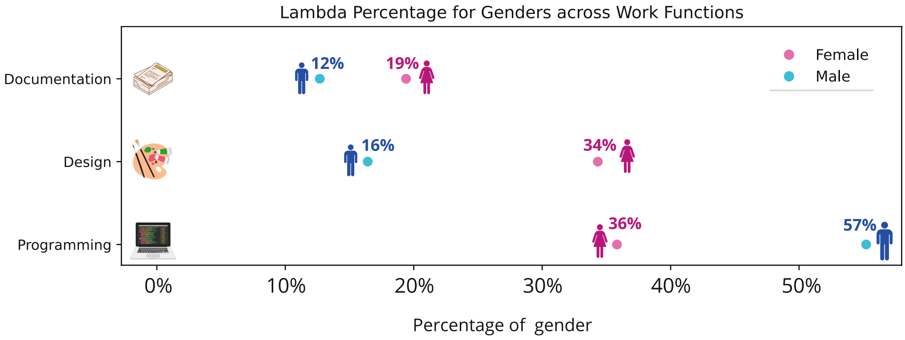


Fig. 3. Primary work function in teamwork, distributed by gender

these results by gender, women design and document more, while men program more. As illustrated in Fig. 3, 34% of women report designing as their primary work function, in comparison to only 16% of all men. This is despite equal amounts of women and men attending the design study program that year. Furthermore, while 36% of all women report programming as their main work function, 56% of all men report programming. Lastly, 19% of women report documentation, while 12% of men primarily document. In total, 89% of the women and 85% of the men either programmed, designed, or documented, the rest distributed between *testing*, *architecture*, and *others*.

4.4 Impact of Gender Composition on Team Dynamics

Changes in Primary Work Functions: Women report on designing to a greater extent than men in all gender compositions, except for in teams with 5 women and 1 man. Teams with 2 women and 4 men represent the team composition where women design the most. 50% of women in those teams report on having design as their primary work function, while only 11% of men in those teams report on primarily designing. We found that women, when in the minority on tilted teams (Fig. 2), often experienced not being able to engage in programming as much as they would have liked. To the open-ended question,"Describe how you experienced the gender distribution in your team, and how you believe it affected the teamwork and the project", one woman in such a tilted team responded:

"There was a tendency for gatekeeping at a technical level by the boys towards the girls. Initially, tasks related to documentation and design were predominantly delegated to the girls, an issue that was addressed and managed somewhat late in the development process." Furthermore, we found that women were expected to design to a greater extent than men. As exemplified by a man as the majority in a tilted team sharing his thoughts about his team's gender composition:

> *"There were two women; I thought they would show significant effort regarding design and delegation, but I ended up doing their job."*

Regarding programming, teams with four or five women have the highest percentage (44%) of women reporting programming as their primary function. Conversely, 22% of the token women in teams with five men reported programming, 25% of the minority women in teams with four men, and 28% of the women in balanced teams. Additionally, the teams of 4 or 5 women exhibit the smallest difference in percentage between women and men primarily involved in programming. One token man in a team with 5 women, suggested that women-dominated teams benefit from a safe exploratory environment:

> *"I believe it had the most positive impact on teamwork because I collaborate well with girls and learned a lot about programming since working with "typical" IT guys has been quite unfamiliar to me. I could ask silly questions while pair programming and maintain a friendly and chatty tone while programming."*

In uniform men teams, 60% of the men program, while only 6% design in these teams. Being the team composition with the largest proportion of members primarily programming and least primarily designing, suggests that men focus less on designing and more on programming.

Teamwork Quality: Table 3 shows each gender composition's average teamwork quality scores. Teams with two women and four men have the lowest mean TWQ scores (3.60), followed by balanced teams (3,76). Teams with five women and one man score the highest (4.37), and teams with four women and two men score the second highest (4.27). It is worth noticing the low standard deviation among all compositions, indicating a high level of agreement among the respondents. However, the two lowest-scoring teams also have the highest standard deviation, suggesting a higher level of disagreement regarding their teamwork quality.

Gender Bias Experiences: Intrigued by the difference in TWQ scores, we performed a correlation analysis to examine the relationship between the number of women in a team and the outcomes of gender-bias post-survey questions. Responses from women and men within the same team were aggregated separately to capture each gender's collective team perspective.

For men, the only significant correlation was with the item, "I am dissatisfied with the gender balance", which yielded a significant negative correlation ($r = -0.553, p < 0.01$). This suggests that their dissatisfaction with gender balance decreases as the number of women in the team increases:

> [Man in a team with one woman and five men]: *"I believe a more balanced gender distribution would have been beneficial for group dynamics."*
> [Man in a team with five women and one man]: *"I experienced it as positive and found it to be enlightening to be in a group with only girls."*

Table 3. TWQ outcomes by gender composition (TWQ factors: communication, coordination, mutual support, effort, cohesion, and balance of member contribution)

Team composition	Comm.	Coord.	Mut. Sup.	Effort	Cohes.	Bal. Contrib.	TWQ Mean	TWQ SD
0W/6M	4.12	4.18	4.42	3.72	4.19	4.15	4.13	0.38
1W/5M	4.18	4.15	4.43	4.05	4.30	4.42	4.25	0.26
2W/4M	3.62	3.41	3.84	3.25	3.67	3.85	3.60	0.52
3W/3M	3.80	3.74	3.90	3.46	3.80	3.85	3.76	0.58
4W/2M	4.25	4.20	4.46	4.09	4.36	4.28	4.27	0.33
5W/1M	4.58	4.08	4.70	3.94	4.44	4.46	4.37	0.48

In contrast, women demonstrated a broader range of significant correlations. Aligning with the men, their dissatisfaction toward the teams' gender balance significantly negatively correlated to the number of women in the team ($r = -0.545, p < 0.01$). Additionally, their feeling of being undervalued by team members was significantly negatively correlated ($r = -0.393, p < 0.01$), while the perceived taboo of discussing gender discrimination also showed a significant negative correlation ($r = -0.401, p < 0.01$). This suggests that as the number of women in a team increases, women tend to be more satisfied with the team's gender balance, feel more valued, and are less likely to see gender discrimination topics as taboo. A woman in a team composed of two women and four men shared her experiences, highlighting potential bias in team interactions:

> *"Some in the group were very inclined to direct the coding questions exclusively to the male programmer, assuming he had the answers and seeking code reviews from him on GitHub. This could be a combination of me not speaking up loudly enough, possibly not appearing confident, and might be a subjective perception."*

Furthermore, their concern about not contributing enough was inversely related to the number of women in the team ($r = -0.241, p < 0.05$), and the TWQ mean score showed a positive correlation ($r = 0.272, p < 0.05$). This indicates that women are less concerned about how their contributions are perceived, and the quality of teamwork improves with a higher representation of women in the team.

These findings indicate a skewed relationship between men's and women's self-reported affection to gender composition. The significant correlations for women across multiple aspects suggest that gender balance within teams could play a critical role in women's team experience, whereas for men's, it appears to be a less critical determinant.

5 Discussion

We have described our strategy for composing agile software engineering student teams, and have examined gender differences in agile teamwork, and the impact of gender composition on student teamwork by forming teams with varied gender compositions.

According to a recent study [5], men are typically more inclined to lead than women, and women are more willing to lead teams with a female majority. Our findings indicate that women on average, participate in the role of Scrum Master to a greater extent than the men. This is in line with Paasivaaras [32] recent study on the Scrum Master role in student teams, who found the Scrum Master role fit female students extremely well and suggests that marketing this role towards girls currently outside the computer science field could be a good idea.

We found that women design and document more than men in the project, while men program more than women. This is consistent with former research, suggesting differences in women's and men's primary work functions to be due to gender stereotypes [1]. However, in light of gender composition within the teams, as the proportion of women in the team increases, we observed a corresponding rise in women engaged in programming.

In general, we observed a positive impact on teamwork quality with an increasing number of women. Women-skewed (4 women/2 men) and female-dominated (5 women/1 man) teams exhibit the highest two TWQ scores, while male-skewed (2 women/4 men) and balanced teams (3 women/3 men) scored the lowest. Where according to Kanter's theory we would expect the teams with token women to have the lowest teamwork quality [19], our results showed these teams to score higher than male-skewed teams, equal teams, and uniform male teams. However, former laboratory experiments, testing the group's gender proportions affection on performance, found a deficiency in women's performance proportionate to the increase of male team members, while men performed unaffected by gender composition [18,40]. Our survey sample lacks significant representation of token women (response rate: 41%), leading to the men's perception of the teamwork dominating these results. Furthermore, supporting the low teamwork quality scores in balanced teams, the recent study by Graßl et al. [12] found that student teams with greater gender diversity encounter challenges in motivation, productivity, respect, and collaboration.

As the number of women on the team increases, we found a decrease in women's reported sense of being undervalued, a reduced taboo surrounding discussions of gender discrimination, and less fear of being perceived as inadequate contributors to team efforts. This gender difference aligns with earlier research, where in relationship to their team members, women felt less respected and appreciated compared to men [12].

The majority of the students implemented daily stand-up meetings and found it to be a valuable practice. Stand-up meetings are often seen as positive, as long as the frequency is adapted, and the meeting is not mainly about reporting progress [30,39]. Recent research [15] has found that implementing social agile practices positively affects psychological safety in the teams. Also, most teams used two communication platforms (the most popular were Discord, Facebook Messenger, Microsoft Teams, and Slack). Slack has been identified as a significant tool that offers various advantages such as facilitating improved communication by reducing the barrier to seeking assistance and encouraging greater participation through the visibility of queries and topics under discussion [34,38]. In student teams, Ross [34] found that Slack seems to improve the quality of group communications and perceived learning outcomes from group work.

5.1 Limitations

Constrained by the limited participation of women at 34%, only two teams were composed of a token man and five women. The small sample size of these teams, challenges drawing definitive conclusions regarding gender composition in women-dominated teams. However, we believe our results provide insight into an infrequently observed team composition within the context of agile software engineering student teams. Furthermore, our data sample of teamwork quality is based on students' independent survey self-assessment, which may not always accurately reflect their team's actual performance.

6 Conclusion

In this study, we investigated gender dynamics within agile software development teams in an educational setting. Our findings reveal that gender composition significantly impacts team collaboration and individual roles within these teams. We conducted two surveys. First, we investigated the ambition level of 240 students (82 women and 158 men) before starting agile project work. When comparing the reported ambition level among genders, we found that women reported a higher average ambition level than men. Next, after the project work, we analyzed a post-survey with 201 respondents. We asked the students to assess teamwork quality, success, performance, self-assessed perception of gender biases, and gender compositions' influence on teamwork. We found that women were more satisfied with stand-up meetings than men. When analyzing team averages with team composition, we found that teams with two women and four men have the lowest score on teamwork quality.

For future research, it would be interesting to observe an all-female team in this context and compare their teamwork experiences to an all-men team. Finally, the course is currently in session, with an increased percentage of women participation at 39%. This facilitates further and deeper research on gender composition in agile student teams. We found, consistent with earlier research [32], that women show a preference for the role of Scrum Master. Given the rising utilization of agile coaches in the industry, and that leadership and project management skills are important traits to have in this role [37], it would be interesting to implement and investigate the role of agile coaches in software engineering capstone courses. Further research could explore whether the skills and inclinations that draw women to the Scrum Master role could similarly influence their participation as agile coaches, potentially enhancing team effectiveness and project outcomes in the IT sector.

Appendix A: Registration of Teams for Spring 2023

The pre-survey with the questions for ambition level is available online: https://doi.org/10.5281/zenodo.10671681.

Appendix B: Teamwork Quality and Gender Balance

The post-teamwork survey is accessible online at: https://doi.org/10.5281/zenodo.10671697.

References

1. Aeby, P., Fong, R., Vukmirovic, M., Isaac, S., Tormey, R.: The impact of gender on engineering students' group work experiences. Int. J. Eng. Educ. **35**(3), 756–765 (2019)
2. Bastarrica, M.C., Perovich, D., Samary, M.M.: What can students get from a software engineering capstone course? In: 2017 IEEE/ACM 39th International Conference on software engineering: software engineering Education and Training Track (ICSE-SEET), pp. 137–145. IEEE (2017)
3. Berntzen, M., Stray, V., Moe, N.B.: Coordination strategies: Managing inter-team coordination challenges in large-scale agile. In: Gregory, P., Lassenius, C., Wang, X., Kruchten, P. (eds.) XP 2021. LNCS, vol. 419, pp. 140–156. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-540-73101-6_9
4. Borges, G.G., de Souza, R.C.G.: Skills development for software engineers: systematic literature review. Inf. Softw. Technol. **168**, 107395 (2024)
5. Born, A., Ranehill, E., Sandberg, A.: Gender and willingness to lead: does the gender composition of teams matter? Rev. Econ. Stat. **104**(2), 259–275 (2022)
6. Braun, V., Clarke, V.: Thematic analysis. American Psychological Association (2012)
7. Canedo, E.D., Tives, H.A., Marioti, M.B., Fagundes, F., de Cerqueira, J.A.S.: Barriers faced by women in software development projects. Information **10**(10), 309 (2019)
8. Cico, O., Jaccheri, L., Nguyen-Duc, A., Zhang, H.: Exploring the intersection between software industry and software engineering education-a systematic mapping of software engineering trends. J. Syst. Softw. **172**, 110736 (2021)
9. Curşeu, P.L., Pluut, H.: Student groups as learning entities: the effect of group diversity and teamwork quality on groups' cognitive complexity. Stud. High. Educ. **38**(1), 87–103 (2013)
10. Dorner, M., et al.: Taxing collaborative software engineering. IEEE Softw. (2023)
11. Dzvonyar, D., Alperowitz, L., Henze, D., Bruegge, B.: Team composition in software engineering project courses. In: Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials, pp. 16–23 (2018)

12. Graßl, I., Krusche, S., Fraser, G.: Diversity and teamwork in student software teams. In: Proceedings of the 5th European Conference on Software Engineering Education, ECSEE 2023, pp. 110-119. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3593663.3593687
13. Happe, L., Buhnova, B.: Frustrations steering women away from software engineering. IEEE Softw. **39**(4), 63–69 (2021)
14. Hashmi, S.I., Markkula, J.: Team composition in software engineering education. In: Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, EASE 2023, pp. 263–264. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3593434.3593464
15. Hennel, P., Rosenkranz, C.: Investigating the "socio" in socio-technical development: the case for psychological safety in agile information systems development. Proj. Manag. J. **52**(1), 11–30 (2021)
16. Hoegl, M., Gemuenden, H.G.: Teamwork quality and the success of innovative projects: a theoretical concept and empirical evidence. Organ. Sci. **12**(4), 435–449 (2001)
17. Iacob, C., Faily, S.: Exploring the gap between the student expectations and the reality of teamwork in undergraduate software engineering group projects. J. Syst. Softw. **157**, 110393 (2019)
18. Inzlicht, M., Ben-Zeev, T.: A threatening intellectual environment: why females are susceptible to experiencing problem-solving deficits in the presence of males. Psychol. Sci. **11**(5), 365–371 (2000)
19. Kanter, R.M.: Some effects of proportions on group life: skewed sex ratios and responses to token women. Am. J. Sociol. **82**(5), 965–990 (1977)
20. Kropp, M., Meier, A.: Collaboration and human factors in software development: teaching agile methodologies based on industrial insight. In: 2016 IEEE Global Engineering Education Conference (EDUCON), pp. 1003–1011. IEEE (2016)
21. Kropp, M., Meier, A., Mateescu, M., Zahn, C.: Teaching and learning agile collaboration. In: 2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T), pp. 139–148. IEEE (2014)
22. Lewis, P.: The quest for invisibility: female entrepreneurs and the masculine norm of entrepreneurship. Gender Work Organ. **13**(5), 453–469 (2006)
23. Lewis, P., Simpson, R.: Kanter revisited: gender, power and (in) visibility. Int. J. Manag. Rev. **14**(2), 141–158 (2012)
24. Lindqvist, A., Sendén, M.G., Renström, E.A.: What is gender, anyway: a review of the options for operationalising gender. Psychol. Sexual. **12**(4), 332–344 (2021)
25. Lindsjørn, Y., Bergersen, G.R., Dingsøyr, T., Sjøberg, D.: Teamwork quality and team performance: exploring differences between small and large agile projects. In: Garbajosa, J., Wang, X., Aguiar, A. (eds.) XP 2018. LNCS, vol. 314, pp. 267–274. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-319-91602-6_19
26. Lindsjørn, Y., Sjøberg, D.I., Dingsøyr, T., Bergersen, G.R., Dybå, T.: Teamwork quality and project success in software development: a survey of agile development teams. J. Syst. Softw. **122**, 274–286 (2016)
27. Løvold, H.H., Lindsjørn, Y., Stray, V.: Forming and assessing student teams in software engineering courses. In: Paasivaara, M., Kruchten, P. (eds.) XP 2020. LNCS, vol. 396, pp. 298–306. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-58858-8_31
28. Mahnic, V.: A capstone course on agile software development using scrum. IEEE Trans. Educ. **55**(1), 99–106 (2011)

29. Main, J.B., Schimpf, C.: The underrepresentation of women in computing fields: a synthesis of literature using a life course perspective. IEEE Trans. Educ. **60**(4), 296–304 (2017)
30. Masood, Z., Hoda, R., Blincoe, K.: Adapting agile practices in university contexts. J. Syst. Softw. **144**, 501–510 (2018)
31. Moe, N.B., Stray, V., Hoda, R.: Trends and updated research agenda for autonomous agile teams: a summary of the second international workshop at xp2019. In: Hoda, R. (ed.) XP 2019. LNCS, vol. 364, pp. 13–19. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_2
32. Paasivaara, M.: Teaching the scrum master role using professional agile coaches and communities of practice. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), pp. 30–39. IEEE (2021)
33. Parker, R., Sangelkar, S., Swenson, M., Ford, J.D.: Launching for success: a review of team formation for capstone design. Int. J. Eng. Educ. **35**(6), 1926–1936 (2019)
34. Ross, S.M.: Slack it to me: complementing LMS with student-centric communications for the millennial/post-millennial student. J. Mark. Educ. **41**(2), 91–108 (2019)
35. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**, 131–164 (2009)
36. da Silva, F.Q., et al.: Team building criteria in software projects: a mix-method replicated study. Inf. Softw. Technol. **55**(7), 1316–1340 (2013)
37. Stray, V., Memon, B., Paruch, L.: A systematic literature review on agile coaching and the role of the agile coach. In: Morisio, M., Torchiano, M., Jedlitschka, A. (eds.) PROFES 2020. LNCS, vol. 12562, pp. 3–19. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64148-1_1
38. Stray, V., Moe, N.B., Vedal, H., Berntzen, M.: Using Objectives and Key Results (OKRs) and slack: a case study of coordination in large-scale distributed agile. In: Proceedings of the 55th Hawaii International Conference on System Sciences, HICSS, p. 10 (2021). http://hdl.handle.net/10125/80225
39. Stray, V.G., Moe, N.B., Dybå, T.: Escalation of commitment: a longitudinal case study of daily meetings. In: Wohlin, C. (ed.) XP 2012. LNCS, vol. 111, pp. 153–167. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30350-0_11
40. Viallon, M.L., Martinot, D.: The effects of solo status on women's and men's success: the moderating role of the performance context. Eur. J. Psychol. Educ. **24**, 191–205 (2009)
41. Wharton, A.S., Baron, J.N.: So happy together? the impact of gender segregation on men at work. Am. Sociol. Rev. 574–587 (1987)
42. Yin, R.K.: Case Study Research: Design and Methods, vol. 5. Sage, Thousand Oaks (2009)

Author Index

A
Abrahamsson, Pekka 117
Alander, Sari 144
Almås, Steffen 179

B
Barbala, Astri 20, 53
Berntzen, Marthe 36, 53

C
Cabrero-Daniel, Beatriz 163
Conboy, Kieran 129

D
Dey, Tapajit 129

E
Eder, Martin 163
Enberg, Tomi 144
Engdal, Silje Alette 36

F
Fitzgerald, Brian 129

G
Gellein, Maja 36
Goisauf, Manuel 117

H
Healy, Robert 129
Herda, Tomas 117, 163

L
Lewzey, Edwin 129
Lindsjørn, Yngve 179

M
Marek, Krzysztof 107
Matthes, Florian 3, 70
Moe, Nils Brede 36, 53

P
Paasivaara, Maria 144
Philipp, Pascal 70
Pichler, Victoria 163
Przybylek, Adam 107

R
Rayhan, Maruf 117

S
Sæter, Gyda Elisa 179
Schmidt, Johannes 3
Stray, Viktoria 20, 53, 179

T
Tobisch, Franziska 3, 70
Trzesicki, Jacek 107

V
van Daalen, Erik 89
van Solingen, Rini 89

W
Weigelt, Karla 70

Z
Zhang, Zheying 117

D. Šmite et al. (Eds.): XP 2024, LNBIP 512, p. 197, 2024.
https://doi.org/10.1007/978-3-031-61154-4