

LNBIP 596

Alfredo Goldman · Sabrina Marczak ·
Graziela Simone Tonin · Jutta Eckstein ·
Xiaofeng Wang · Renato Cordeiro Ferreira ·
Andrea Pinna · Michael Neumann (Eds.)

Agile Processes in Software Engineering and Extreme Programming – Workshops

XP 2026 Workshops
São Paulo, Brazil, April 8–11, 2026
Revised Selected Papers

Lecture Notes

in Business Information Processing

596

Series Editors

Wil van der Aalst , *RWTH Aachen University, Aachen, Germany*

Sudha Ram , *University of Arizona, Tucson, USA*

Michael Rosemann , *Queensland University of Technology, Brisbane, Australia*

Clemens Szyperski, *Oracle Cloud, Austin, TX, USA*

Giancarlo Guizzardi , *University of Twente, Enschede, The Netherlands*

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.

Alfredo Goldman · Sabrina Marczak ·
Graziela Simone Tonin · Jutta Eckstein ·
Xiaofeng Wang · Renato Cordeiro Ferreira ·
Andrea Pinna · Michael Neumann
Editors

Agile Processes in Software Engineering and Extreme Programming – Workshops

XP 2026 Workshops

São Paulo, Brazil, April 8–11, 2026

Revised Selected Papers

Editors

Alfredo Goldman 
University of São Paulo
São Paulo, São Paulo, Brazil

Graziela Simone Tonin 
INSPER
São Paulo, Brazil

Xiaofeng Wang 
LUT University
Lappeenranta, Finland

Andrea Pinna 
University of Cagliari
Cagliari, Italy

Sabrina Marczak 
Pontifical Catholic University of Rio Grande
do Sul
Partenon - Porto Alegre, Rio Grande do Sul,
Brazil

Jutta Eckstein 
Independent
Braunschweig, Niedersachsen, Germany

Renato Cordeiro Ferreira 
Tilburg University
Tilburg, The Netherlands

Michael Neumann 
University of Applied Sciences and Arts
Hannover
Hannover, Niedersachsen, Germany



ISSN 1865-1348

ISSN 1865-1356 (electronic)

Lecture Notes in Business Information Processing

ISBN 978-3-032-39204-6

ISBN 978-3-032-39205-3 (eBook)

<https://doi.org/10.1007/978-3-032-39205-3>

© The Editor(s) (if applicable) and The Author(s) 2027. This book is an open access publication.

Open Access This book is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this book are included in the book's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the book's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

Unless otherwise expressly agreed by the Publisher in writing, the Publisher does not authorize use or reproduction of all or any part of this work in any manner for the purpose of training, fine-tuning, evaluating, or developing artificial intelligence models, technologies, or systems. In accordance with Article 4(3) of the EU Directive 2019/790 on copyright and related rights in the Digital Single Market, the Publisher expressly reserves all rights with respect to this work under the text and data mining exception of the directive.

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: Gewerbstrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Preface

With great pleasure we introduce this volume of research proceedings from the materials of the research workshops and the Posters track of the 27th International Conference on Agile Software Development, XP 2026, held during April 8–11, 2026, at the INSPER-Institute of Education and Research, São Paulo, Brazil

Research papers from the XP 2026 conference were published in the main conference proceedings, LNBIP volume 578. This companion volume, published after the conference, contains selected and revised scientific papers as well as summaries from four scientific workshops, the Agile Education and Training track, the Agile Software Engineering Education in the Age of AI: Rethinking Practices, Pedagogies, and Tools track, and the Posters track.

The main ambition with the workshops was to foster collaboration, generate new insights, and advance the field of agile software development. In total, we received seven proposals for research workshops, of which five were accepted. The selected workshops were all characterised by a high degree of interactivity, and had potential to attract a good number of participants on the day before the main conference.

The final research workshop program consisted of five half-day workshops. They each gathered the interest of from 14–34 participants. This volume includes the selected research papers from four of these workshops.

- 2nd Agile Practice & Research (PR) Workshop
- The Fourth International Workshop on Global and Hybrid Work in Software Engineering (GoHyb)
- Human-AI Collaboration in Agile Teams
- International Workshop on AI for Technical Debt Management in Agile Software Development (AI4TD)

Out of a total of 35 submissions, 18 research papers were submitted to the workshops, of which 11 were accepted for publication in the post-proceedings.

This year, the volume also features selected and revised research papers from the track “Agile Software Engineering Education in the Age of AI: Rethinking Practices, Pedagogies, and Tools.” A total of five papers were submitted to this track, of which four were accepted for publication.

The research poster track culminated with an exhibition held on April 9 and included 3 posters, out of which 1 was accepted for publication.

All the tracks listed above followed a single-blind review process in EasyChair, with each submission evaluated by at least three reviewers.

We express our sincere appreciation to the authors featured in this volume for their contributions to advancing the field of agile software development. We also thank the workshop organizers for their commitment to encouraging collaboration and progress in this area. Our thanks go as well to the reviewers for their careful evaluations and constructive feedback, which contributed to the quality of the papers.

We further recognize the important role of all speakers, sponsors, shepherds, chairs, and volunteers in making this conference a success. Special appreciation is extended to the XP Conference Steering Committee and to our supporters: the sponsor Google, the host INSPER, and our partners IME-USP and PUCRS.

June 2026

Alfredo Goldman
Sabrina Marczak
Graziela Simone Tonin
Jutta Eckstein
Xiaofeng Wang
Renato Cordeiro Ferreira
Andrea Pinna
Michael Neumann

Organization

General Chairs

Alfredo Goldman
Sabrina Marczak

University of São Paulo, Brazil
Pontifical Catholic University of Rio Grande do
Sul, Brazil

General Co-chairs

Alfredo Goldman
Sabrina Marczak

INSPER, Brazil
Agile Trends, Brazil

Workshops Co-chairs

Jutta Eckstein
Xiaofeng Wang

Independent, Germany
LUT University, Finland

Posters Track Co-chairs

Renato Cordeiro Ferreira
Thatiane de Oliveira Rosa

University of São Paulo, Brazil/Eindhoven
University of Technology, Netherlands
Federal Institute of Tocantins (IFTO), Brazil

Publication Co-chairs

Andrea Pinna
Michael Neumann

University of Cagliari, Italy
University of Applied Sciences Hannover,
Germany

Workshops Organizers

2nd Agile Practice and Research (PR) Workshop

Karen Eilers	BSP Business and Law School (Hamburg), Germany
Michael Neumann	University of Applied Sciences & Arts Hannover, Germany
Eva-Maria Schön	University of Applied Sciences Emden/Leer, Germany
Mali Senapathi	Auckland University of Technology, New Zealand
Maria Rauschenberger	University of Applied Sciences Emden/Leer, Germany
Tiago Silva da Silva	Federal University of São Paulo, Brazil

The Fourth International Workshop on Global and Hybrid Work in Software Engineering (GoHyb)

Maria Paasivaara	LUT University, Finland and Aalto University, Finland
Fatemeh Broomandi	LUT University, Finland

Human-AI Collaboration in Agile Teams

Nils Brede Moe	SINTEF, Norway
Viktoría Stray	University of Oslo/SINTEF, Norway

International Workshop on AI for Technical Debt Management in Agile Software Development

Antonio Martini	University of Oslo, Norway
Phu Nguyen	SINTEF, Norway
Lina Ochoa	Eindhoven University of Technology, Netherlands
Jan Henrik Gundelsby	Knowit Solutions Norway, Norway
Mili Orucevic	Visma, Norway

Agile Training and Education Track

Agile Software Engineering Education in the Age of AI

Philipp Diebold	Bagilstein GmbH & IU International University of Applied Sciences, Germany
Maria Paasivaara	LUT University & Aalto University, Finland

Program Committee

Ademar Aguiar	INESC TEC, Universidade do Porto, Portugal
Adam Przybylek	University of Galway, Ireland
Alessandra Parziale	Gran Sasso Science Institute, Italy
Andrea Capiluppi	University of Groningen, Netherlands
Andrea Janes	Free University of Bozen-Bolzano, Italy
Andrea Pinna	University of Cagliari, Italy
Andreas Hinderks	University of Applied Sciences and Arts Hannover, Germany
Anh Nguyen Duc	University of South Eastern Norway, Norway
Antonio Martini	University of Oslo, Norway
Arne Løvold	Statens pensjonskasse, Norway
Astri Barbala	SINTEF Digital, Norway and University of Galway, Ireland
Eduardo Guerra	Free University of Bozen-Bolzano, Italy
Eva-Maria Schön	University of Applied Sciences Emden/Leer, Germany
Fatemeh Broomandi	LUT University, Finland
Iflaah Salman	Lappeenranta-Lahti University of Technology, Finland
Jan Henrik Gundelsby	Knowit Solutions Norway, Norway
Javier Gonzalez Huerta	Sweden
Jens Heidrich	Hochschule Mainz - University of Applied Sciences, Germany
Jingyue Li	Norwegian University of Science and Technology, Norway
João Paulo Biazotto	University of Groningen, Netherlands and University of São Paulo, Brazil
Julian Bass	University of Salford, UK
Jutta Eckstein	Independent, Germany
Karen Eilers	BSP Business and Law School (Hamburg), Germany

Leevi Rantala	University of Oulu, Finland
Leonard Peter Binamungu	University of Dar es Salaam, Tanzania
Leonhard Faubel-Teich	University of Hildesheim, Germany
Lina Ochoa	Eindhoven University of Technology, Netherlands
Maria Paasivaara	LUT University and Aalto University, Finland
Maria Rauschenberger	University of Applied Sciences Emden/Leer, Germany
Marius Mikalsen	Norwegian University of Science and Technology and SINTEF, Norway
Martin Kropp	University of Applied Sciences and Arts Northwestern, Switzerland, Switzerland
Marthe Berntzen	University of Oslo, Norway
Michael Neumann	University of Applied Sciences & Arts Hannover, Germany
Mili Orucevic	Visma, Norway
Nils Brede Moe	SINTEF, Norway
Paul Clarke	Dublin City University and Lero, Ireland
Peggy Gregory	University of Glasgow, UK
Phu Nguyen	SINTEF, Norway
Philipp Diebold	IU International University of Applied Sciences, Germany
Rafael Capilla	Universidad Rey Juan Carlos, Spain
Sibylle Peter	University of Applied Sciences and Arts Northwestern, Switzerland, Switzerland
Stefen Fraser	Innoxec, USA
Thatiane de Oliveira Rosa	Federal Institute of Tocantins, Brazil
Tomas Herda	Austrian Post Group IT, Austria
Torgeir Dingsøy	Norwegian University of Science and Technology and SimulaMet, Norway
Victoria Pichler	Austrian Post Group IT, Austria
Wardah Awan	LUT University, Finland
Yen Ying Ng	Nicolaus Copernicus University in Toruń, Poland
Yngve Lindsjörn	Oslo Metropolitan University, Norway
Zadia Codabux	University of Saskatchewan, Canada

Steering Committee

Peggy Gregory (Chair)	University of Glasgow, UK
Hubert Baumeister	Technical University of Denmark, Denmark
François Coallier	École de Technologie Supérieure, Canada
Jutta Eckstein	Independent, Germany

Hendrik Esser	Ericsson, Germany
Juan Garbajosa	Universidad Politécnica de Madrid, Spain
Martin Kropp	University of Applied Sciences and Arts, Northwestern Switzerland
Wouter Lagerweij	Lagerweij Consultancy, Netherlands
Maria Paasivaara	LUT University & Aalto University, Norway
Viktoria Stray	University of Oslo, Norway
Graziela Simone Tonin	INSPER, Brazil
Xiaofeng Wang	LUT University, Finland

Sponsoring Organizations

Google (Sponsor)
INSPER (Hosting)
IME-USP (Partner)
PUCRS (Partner)

Contents

2nd Agile Practice and Research (PR) Workshop

The 2nd Workshop on Agile Practice and Research: A Summary and Call For Research	3
<i>Karen Eilers, Michael Neumann, Eva-Maria Schön, Mali Senapathi, Maria Rauschenberger, and Tiago Silva da Silva</i>	
From Education to Evidence: A Collaborative Practice Research Platform for AI-Integrated Agile Development	13
<i>Tobias Geger, Andreas Rausch, Ina Schiering, Frauke Stenzel, and Stefan Wittek</i>	
Aligning Research and Product Development Through Multi-Cycle Design Science Research: An Experience Report	22
<i>Alexandre Nodari, Marcelo Luis Walter, Leonardo Nunes, Marcelo E. Pellenz, and Edson Emilio Scalabrin</i>	
Designing Transfer-Ready Research Artefacts for Ways of Working	31
<i>Philipp Diebold</i>	
Closing Research–Practice Gaps through Action Research: Empowering Teams to Strengthen their Team Effectiveness	40
<i>Anna Rook, Bente Schreiter, and Christiaan Verwijs</i>	

Human-AI Collaboration in Agile Teams

AI-Augmented Agile Teams: A Research Agenda and Summary of the Human-AI Collaboration Workshop	51
<i>Nils Brede Moe and Viktoria Stray</i>	
“I’ll Ask AI First”: How GenAI Disrupts Collaboration in Agile Teams	61
<i>Jan Henrik Gundelsby, Marthe Berntzen, Camilla Grøneng, and Dan Vigeland</i>	
From AI Assistants to AI-Dev-Systems: A Three-Layer Model for Human-AI Collaboration in Agile Software Development	71
<i>Andreas Hinderks, Jörg Thomaschewski, and Eva-Maria Schön</i>	

Human–AI Collaboration for Scaling Agile Regression Testing:
An Agentic-AI Teammate from Manual to Automated Testing 80
*Moustapha El Outmani, Manthan Venkataramana Shenoy,
Ahmad Hatahet, Andreas Rausch, Tim Niklas Kniep, Thomas Raddatz,
and Benjamin King*

**International Workshop on AI for Technical Debt Management in
Agile Software Development**

AI for Technical Debt Management: Summary from the AI4TD Workshop
at XP2026 91
*Antonio Martini, Phu Nguyen, Lina Ochoa, Jan Henrik Gundelsby,
and Mili Orucevic*

From Smell to Solution: Multi-agent LLMs Analyzing Cyclic
Dependencies for LLM-Enabled Refactoring 100
*Henrik Brunvatne Olafsen, Adela Nedisan Videsjorden, Arda Goknil,
and Sagar Sen*

Reducing Technical Debt with Generative AI: A Case Study of AI-Assisted
Refactoring 111
Iselin Mordal Bakke, Viktoria Stray, and Yngve Lindsjørn

**The Fourth International Workshop on Global and Hybrid Work in
Software Engineering (GoHyb)**

Exploring the Role of AI in Hybrid Work Within Software Engineering 123
Fateme Broomandi and Maria Paasivaara

High Trust and High Flexibility: Hybrid Work Practices in the Norwegian
Software Industry 133
Nils Brede Moe, Viktoria Stray, Maria Hope, and Mathias Willumsen

Productivity and Collaboration in Hybrid Agile Teams: An Interview Study 143
Elisabeth Mo, Jefferson Seide Molléri, and Asle Fagerstrøm

**Agile Software Engineering Education in the Age of AI: Rethinking
Practices, Pedagogies, and Tools**

Preparing Students for AI-Driven Agile Development: A Project-Based
AI Engineering Curriculum 155
Andreas Rausch, Stefan Wittek, Tobias Geger, and David Inkermann

Teaching Agile Requirements Engineering: A Stakeholder Simulation with Generative AI	162
<i>Eva-Maria Schön, Michael Neumann, and Tiago Silva da Silva</i>	
Introducing Agile Team Practices in Capstone Courses: Impact on Student Collaboration	171
<i>Viktoria Stray, Yngve Lindsjörn, Idunn Stabell, Camilla Kielland Lund, Steffen Almås, and Marthe Berntzen</i>	
Teaching Agile Methods in K-12 Computer Science Education: An Experience Report on Preparing Teachers to Coach in Classrooms	181
<i>Konstantin Oltmann and Ralf Romeike</i>	
Posters Track	
Making Technical Debt Visible: An Indicator-Driven Radar for Agile Teams ...	193
<i>Mateus Costa Lucena, Antonio Carlos Pacagnella Junior, and Gizelle Sandrini Lemos</i>	
Author Index	199

2nd Agile Practice and Research (PR) Workshop



The 2nd Workshop on Agile Practice and Research: A Summary and Call For Research

Karen Eilers¹ , Michael Neumann² , Eva-Maria Schön³ ,
Mali Senapathi⁴ , Maria Rauschenberger³ , and Tiago Silva da Silva⁵ 

¹ BSP Business and Law School - Campus Hamburg, Hamburg, Germany
karen.eilers@bsp-campus-hamburg.de

² University of Applied Sciences and Arts Hannover, Hannover, Germany
michael.neumann@hs-hannover.de

³ University of Applied Sciences Emden/Leer, Emden/Leer, Germany
{firstname.lastname,firstname.lastname}@hs-emden-leer.de

⁴ Auckland University of Technology, Auckland, New Zealand
mali.senapathi@aut.ac.nz

⁵ Federal University of São Paulo, São Paulo, Brazil
silva.tiago@unifesp.br

Abstract. Agile software development has been shaped by the interplay between academic research and industrial practice for over two decades, yet notable gaps persist between both domains. This paper focuses on three research-practice gaps: the theory gap, the time gap, and the transfer gap. To address these, the *2nd Agile Practice & Research Workshop* was held at the International Conference on Agile Software Development (XP) 2026 in São Paulo, Brazil, bringing researchers and practitioners together to identify root causes and develop joint solutions. Building on two preceding sessions in which contributions of participants had been presented, participants engaged in a structured collaborative session, working in small groups on one of the three gaps and reflecting on possible causes and remedies. The organizers synthesized the results into four propositions for improving the research-practice intersection: (1) improving scientific communication, (2) aligning research more closely with emerging industrial needs, (3) creating stronger incentives for sustained collaboration, and (4) integrating educational approaches into research practice. From these, three calls for research were formulated: (a) broader adoption of open science practices for transparency, reproducibility, and cumulative evidence; (b) higher empirical quality standards through stronger theoretical grounding and rigorous design; and (c) more explicit, value-oriented contributions that clearly articulate their practical and scientific relevance. The paper offers both a summary of the workshop and a call to strengthen research-practice collaboration.

Keywords: agile software development · agile practice & research · empirical software engineering · open science · workshop summary

1 Introduction

Agile software development has attracted sustained attention in both research and practice for more than two decades [1]. During this time, a substantial body of knowledge has emerged on a wide range of topics, including agile methods and practices, scaling, requirements engineering, and human-centered software development. At the same time, agile practice has continuously evolved in response to changing organizational, technological, and societal conditions [8]. This ongoing evolution has increased both the need for and the difficulty of maintaining a strong connection between academic research and industrial practice.

Despite the maturity of the field, important gaps remain between agile research and practice. Practitioners often perceive research findings as difficult to access, insufficiently actionable, or only partially relevant to the complexity of real-world settings. At the same time, academic research frequently struggles to keep pace with the rapid development of industrial practices and emerging challenges [2]. To better understand and address this tension, it is useful to distinguish between different dimensions of the research-practice gaps in agile software development.

To contribute to this discussion, the *2nd Agile Practice & Research Workshop* was organized as part of the *International Conference on Agile Software Development (XP)2026*. The workshop provided a platform for researchers and practitioners to share perspectives, discuss current challenges, and jointly reflect on how the relationship between agile research and practice can be strengthened. This paper summarizes the workshop and its main outcomes.

In this paper, we focus on three particularly relevant gaps that continue to shape the relationship between agile research and practice.

First, the *theory gap* refers to the limited theoretical grounding of much agile research. Research in the agile community is still dominated by empirical case studies, which is common in interdisciplinary fields such as agile software development. However, practitioners often question the external validity of such studies and whether their findings can be generalized across the broader industry [19]. In addition, literature [21] points to theoretical limitations, such as a strong emphasis on framework practices rather than the underlying principles of the *Agile Manifesto* [3]. Although agile transformations and related phenomena have been widely studied (*e.g.*, [4,6]), there is still a need for stronger theoretical development that can explain, guide, and sustain agile practices across contexts [11,16].

Second, the *time gap* describes the challenge that industrial practice often evolves more rapidly than academic research. Over the past two decades, agile ways of working have continuously adapted to new organizational demands, technological developments, and shifting work environments [8]. Since 2020, this gap has become even more pronounced due to accelerated disruptive changes, including remote and hybrid work as well as the increasing integration of Artificial Intelligence (AI) into software development processes [14]. While agile teams in practice often show considerable resilience in responding to such developments, research frequently struggles to keep pace and to actively shape emerging trends.

Addressing this imbalance requires closer alignment between the timescales of research and the realities of industry [2].

Third, the *transfer gap* captures the difficulty of making academic insights accessible, understandable, and actionable for practitioners. Although research generates valuable knowledge, its findings are often not communicated in ways that directly support decision-making and application in practice. Differences in language, priorities, and timescales between academia and industry further reinforce this challenge. Bridging the transfer gap, therefore, requires stronger collaboration between researchers and practitioners, as well as formats that translate research findings into applicable guidance for real-world agile settings [19, 20]. Together, these three gaps underscore the need for stronger academia-industry collaboration and form the conceptual foundation of the workshop summarized in this paper.

The paper is structured as follows. Section 2 outlines the workshop organization and thematic structure, followed by the results of the collaborative discussions. In Sect. 3, we present our propositions to close the gaps before this summary closes with the calls for future research in Sect. 4.

2 Workshop Organization

Building on the results and findings of the first edition of the workshop in 2025 [13], the second edition aimed to further strengthen the dialog between academia and industry. The workshop was designed as a half-day event within XP 2026 on 08.April 2026 in São Paulo, Brazil. Based on the three gaps, we invited contributions from both researchers and practitioners that addressed challenges and opportunities at the intersection of agile practice and research. This year, we brought together 20 researchers and practitioners with diverse backgrounds, from Master’s and PhD students to professors, and from consultants to software engineers and developers.

2.1 Session 1: Bridging Time and Transfer Gaps Between Research and Practice

The first session focused on the challenges of aligning academic research with the pace and needs of industrial practice. Diebold [5] addressed this issue in his contribution *Designing Transfer-Ready Research Artifacts for Ways of Working*, which discussed how research-based insights can be translated into practitioner-oriented artifacts that remain theoretically grounded while becoming easier to apply in real-world contexts. Walter et al. [18] further explored this challenge in *Aligning Research and Product Development through Multi-cycle Design Science Research: An Experience Report*, presenting a model for synchronizing academic research cycles with product development needs in industry. Contributions in this session therefore focused on solutions on how research can remain relevant in rapidly changing agile environments and how academic knowledge can be communicated in ways that are more accessible and actionable for practitioners.

The discussion highlighted the need for more timely research, stronger exchange formats between academia and industry, and better mechanisms for translating findings into practical guidance.

2.2 Session 2: Grounding Research for Practice: Closing the Theory Gap

The second session addressed the theory gap in agile software development research. Geger et al. [9] contributed with *From Education to Evidence: A Collaborative Practice Research Platform for AI-Integrated Agile Development*, showing how project-based agile education can serve as a collaborative research platform for generating timely, practice-relevant evidence. Rook et al. [15] complemented this perspective with *Closing Research–Practice Gaps Through Action Research: Empowering Teams to Strengthen Their Team Effectiveness*, which demonstrated how action research and theory-informed feedback loops can support teams in improving their ways of working while contributing to empirical theory development. Contributions in this session emphasized the importance of stronger theoretical grounding in order to move beyond isolated empirical observations and generate knowledge that is transferable across contexts. The discussion also raised the question of how theory development in agile research can be better connected to the realities of practice, so that conceptual advances remain both scientifically rigorous and practically relevant.

2.3 Collaborative Session

The collaborative session was designed to move beyond the presentation of individual contributions and to foster a structured discussion on how the identified gaps between agile research and practice can be addressed. To this end, the participants were divided into three small groups. To form the groups, the three tables at which the participants were initially seated were used as a starting point. Some participants were then asked to join a different group in order to ensure diversity with regard to participant characteristics (e.g., gender, age, and practitioner or researcher background) as well as to enable balanced group sizes.

Each group was assigned one of the three previously identified gaps, namely the theory gap, the time gap, or the transfer gap. During the group work phases, the groups self-organized and documented the results of their discussions on sticky notes. The collaborative session comprised three small-group activities (15-minute timeboxes each) covering the following topics:

(a) *root causes of the gaps*: each group was asked to discuss and identify potential root causes underlying its assigned gap. Participants were encouraged not only to name possible causes but also to justify and critically reflect on them based on their own experiences from research and practice.

(b) *solution ideas to address the gaps*: the groups were invited to develop concrete solution ideas aimed at addressing the identified root causes and, consequently, contributing to closing the respective gap.

(c) *15% solutions*: each group identified measures that they could implement immediately on their own and without additional support or resources to realize the solutions developed in activity b). For this purpose, a method from Liberating Structures was applied.

The time between the activities was used for plenary discussions of the individual group results, allowing participants to compare perspectives across groups and relate the proposed solutions to the broader challenges at the intersection of agile practice and research. For this purpose, the whiteboard walls in the workshop room were used to enable cross-group documentation of the results.

2.4 Synthesis of the Workshop Results

After the workshop, the insights generated during the collaborative session were synthesized by the attending workshop organizers (Author 1 and Author 2 of this summary). The synthesis was conducted immediately after the workshop in the same room. This enabled the documented results of the workshop participants to be used directly as input since they were still displayed on the whiteboard walls. In the first step, the identified root causes (activity a) and proposed solution ideas (activity b) were thematically categorized. In the second step, relationships between root causes and solution ideas were systematically analyzed in order to identify recurring patterns and overarching connections. Based on this synthesis, we derived a set of propositions aimed at improving the intersection between agile research and practice. In addition, we formulated specific calls for researchers intended to encourage the software engineering (SE) community to actively contribute to closing the identified gaps.

3 Propositions for Optimizing the Intersection Between Research & Practice in Agile Software Development

Based on the presented contributions, discussions conducted during the collaborative session and the subsequent synthesis of results, we derived four propositions to improve the intersection between research and practice in agile software development. These propositions reflect recurring themes across the identified root causes and solution ideas and point to areas in which more systematic change is needed.

Proposition 1: Scientific Communication — A first proposition concerns the communication of scientific knowledge beyond academic audiences. A key barrier between research and practice is that practitioners often struggle to understand and apply research findings. To address this issue, scientific communication skills should be more explicitly integrated into PhD study programs and other forms of research training. Researchers should be better prepared to present their work not only in academic settings but also in practitioner-oriented environments such as industry conferences, meetups, and community events. By doing so, researchers learn to identify the similarities between their own motivations

and those of practitioners. In addition, the SE community should experiment with alternative dissemination formats, including podcasts, vlogs, blogs, or similar channels. These formats could include, for example, new research results in practical language, instructional videos demonstrating how science and practice can collaborate, or success stories of previous collaborations in order to communicate evidence-based findings more broadly, more quickly, and more efficiently.

Proposition 2: Value the Momentum — A second proposition emphasizes the importance of aligning research more closely with the topics and dynamics currently shaping industrial practice. To achieve stronger relevance, researchers need a greater awareness of those issues that practitioners perceive as urgent and valuable. In this context, the notion of value plays a central role. Research should more explicitly address practical concerns such as return on investment, business value, process-related pain points, and regulatory challenges. Furthermore, the discussions highlighted that strong networks in practice are essential for recognizing emerging topics early and understanding their relevance. Building and maintaining such networks can therefore help researchers engage with practice at the right time and with greater impact.

Proposition 3: Emphasize Incentives Between Research & Practice — The third proposition addresses the need for stronger incentives that encourage exchange and contribution between research and practice. One challenge lies in the absence of sufficiently visible and attractive mechanisms that motivate both communities to engage with one another. For practitioners, collaboration with academia may create opportunities to increase internal visibility, strengthen their influence within organizations, and contribute practical ideas to the broader scientific discourse. For researchers, closer interaction with practice can improve the relevance, visibility, and applicability of their work. Thus, more explicit incentive structures are needed that make the benefits of collaboration tangible for both sides and encourage sustained engagement across the research–practice divide.

Proposition 4: Integrating Educational Approaches into Research Practice — The fourth proposition focuses on the integration of educational concepts into research practice. The workshop discussions suggested that state-of-the-art educational methods, such as project-based learning, may provide valuable inspiration for the design and execution of research projects. Applying such approaches could help to structure research work more effectively, foster iterative learning, and sharpen the focus of project outcomes. In particular, this may support researchers in narrowing down research contributions and increasing the clarity and usefulness of outputs. Educational integration can therefore be understood as a way to strengthen both the process and the outcome orientation of research in contexts where relevance to practice is essential.

4 Conclusion & Call for Research

By synthesizing the results of this workshop, we also came up with specific calls for other researchers aiming to close the identified gaps:

Call 1: Establish a mandatory Open Science paradigm: To be able to increase transparency and thus understandability of research contributions, we call for a mandatory Open Science paradigm. Under the Open Science paradigm, we understand in accordance with Mendez et al. [12] making research artifacts available to the public. The paradigm further expands to the openness of datasets themselves, data collection and analysis documentation, or open access, which can be ensured by publishing accepted papers as pre-prints. As the discussion on open science in SE goes back more than half a decade, we see the need to push this paradigm much further forward. Even if high quality SE venues like the *International Conference on Software Engineering (ICSE)* or the *International Symposium on Empirical Software Engineering and Measurement (ESEM)* have adopted mandatory Open Science policies, we notice that the majority of SE venues have not followed suit yet. We are aware of upcoming challenges and problems, *e.g.*, regarding confidential datasets and open questions on how to deal with losing important contributions by enabling mandatory open science. But we see the need to move more into the direction of replicability and reproducibility of studies in the various different contexts to increase the chain of evidence in empirical SE research.

Call 2: Embrace the empirical research quality criteria towards high standards of rigorous, replicable and theory grounded research: Following-up on the initial call above, we want to engage other researchers to focus even more on empirical research quality criteria and to push their own approaches more towards so-called gold-standard principles. We mean by that not only following standardized or established guidelines for preparing, conducting, and reporting research results. We further want to point to specific research quality criteria. Empirical SE research should be designed by a grounding theory. An analysis by Schmid [16] shows that empirical research focuses mostly on analyzing phenomena while not considering a theoretical grounding. By respecting quality criteria and grounding empirical SE research on a theory, we aim for replicable or reproducible research. In summary, this is what we call a gold standard for empirical SE research.

Call 3: Towards more explicit research contributions: Finally, we want to point out the very important facet of research contributions. The focus on contribution is typically grounded by providing arguments for expanding the existing body of knowledge with valuable additional findings. Although researchers are aware of this important aspect related to their research, it typically lacks one very specific point: What is the value of the results and/or findings? Furthermore, from a so called research gold standard claim, one has to consider that for research preparation and conduction in a systematic manner, the focus on contributions remains lacking. For example, Lago et al. [10] analyzed 91 ICSE 'best' papers between 2013 and 2024 and showed in their study that conclusion validity is the least considered threat in these studies.

Interestingly, these findings show that we are not only challenged by the question of how we should explain or state such contributions but also how to critically reflect on these aspects as part of the limitations.

To summarize, from our point of view, research contributions are mainly divided into two categories: a) Artifacts grounded by research findings or results and b) Artifacts grounded in research methodologies. To be more precise, we provide two specific examples. As an example of an artifact grounded in research findings, we refer to the framework “*Genau Meine Arbeitsweise*” [5] discussed by the workshop participants. The value of this framework lies in enabling practitioners to reflect on their own ways of working and to adapt them in a more sustainable and deliberate manner. In this sense, the artifact translates prior empirical findings on work practices, reflection, and organizational change into an applicable instrument for practice. An example of an artifact grounded in research methodologies is the platform “*Columinity.com*” [17], presented and discussed in the workshop. During its use, the platform supports the systematic collection of scientific data while simultaneously guiding teams in applying Action Research principles in their own organizational context. Hence, the artifact creates value not only through its practical functionality but also by embedding a rigorous research methodology into real-world practice and thereby closing research-practice gaps. We provide additional material from the workshop in our git repository [7].

References

1. Baham, C., Hirschheim, R.: Issues, challenges, and a proposed theoretical core of agile software development research. *Inf. Syst. J.* **32**(1), 103–129 (2022). <https://doi.org/10.1111/isj.12336>
2. Barroca, L., Sharp, H., Salah, D., Taylor, K., Gregory, P.: Bridging the gap between research and agile practice: an evolutionary model. *Int. J. Syst. Assur. Eng. Manage.* **9**, 323–334 (2018). <https://doi.org/10.1007/s13198-015-0355-5>
3. Beck, K., et al.: Agile manifesto (2021). <https://agilemanifesto.org/>
4. Carroll, N., Conboy, K., Wang, X.: From transformation to normalisation: an exploratory study of a large-scale agile transformation. *J. Inf. Technol.* **38**(3), 267–303 (2023). <https://doi.org/10.1177/02683962231164428>
5. Diebold, P.: Designing transfer-ready research artefacts for ways of working. In: *Agile Processes in Software Engineering and Extreme Programming - Workshops*, Springer Nature Switzerland, Cham (2026)
6. Dingsoeyr, T., Falessi, D., Power, K.: Agile development at scale: the next frontier. *IEEE Softw.* **36**(2), 30–38 (2019). <https://doi.org/10.1109/MS.2018.2884884>
7. Eilers, K., Neumann, M., Schön, E.M., Senapathi, M., Rauschenberger, M., da Silva, T.S.: Additional material to workshop agilepr at xp2026 (2025). <https://bit.ly/xp2026-agilepr2>
8. Eilers, K., Peters, C., Leimeister, J.M.: Why the agile mindset matters. *Technol. Forecast. Soc. Chang.* **179**, 121650 (2022). <https://doi.org/10.1016/j.techfore.2022.121650>
9. Geger, T., Rausch, A., Schiering, I., Stenzel, F., Wittek, S.: From education to evidence: a collaborative practice research platform for AI-integrated agile development. In: *Agile Processes in Software Engineering and Extreme Programming - Workshops*, Springer Nature Switzerland, Cham (2026)

10. Lago, P., Runeson, P., Song, Q., Verdecchia, R.: Threats to validity in software engineering – hypocritical paper section or essential analysis? In: Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 314–324. ESEM '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3674805.3686691>
11. Magistretti, S., Trabucchi, D.: Agile-as-a-tool and agile-as-a-culture: a comprehensive review of agile approaches adopting contingency and configuration theories. *RMS* **19**, 223–253 (2024). <https://doi.org/10.1007/s11846-024-00745-1>
12. Mendez, D., Graziotin, D., Wagner, S., Seibold, H.: Open Science in Software Engineering, pp. 477–501. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-32489-6_17
13. Neumann, M., Schön, E.M., Senapathi, M., Rauschenberger, M., Silva da Silva, T.: Towards a closer collaboration between practice and research in agile software development workshop: A summary and research agenda. In: Peter, S., et al. (eds.) *Agile Processes in Software Engineering and Extreme Programming – Workshops*, pp. 91–97. Springer Nature Switzerland, Cham (2026)
14. Planötscher, D., Wang, X., Adil, M., Gregory, P., Larimian, S.: Human-AI collaboration in software development activities: perspectives of agile practitioners. In: *Agile Processes in Software Engineering and Extreme Programming*, pp. 274–289. Springer Nature Switzerland, Cham (2026)
15. Rook, A., Schreiter, B., Verwijs, C.: Closing research-practice gaps through action research: Empowering teams to strengthen their team effectiveness. In: *Agile Processes in Software Engineering and Extreme Programming - Workshops*, Springer Nature Switzerland, Cham (2026)
16. Schmid, K.: If you want better empirical research, value your theory: on the importance of strong theories for progress in empirical software engineering research. In: *Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering*, pp. 359–364. Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3463274.3463360>
17. Verwijs, C., Russo, D.: A theory of scrum team effectiveness. *ACM Trans. Softw. Eng. Methodol* **32**(3) (2023). <https://doi.org/10.1145/3571849>
18. Walter, M.L., Nodari, A., Nunes, L., Pellenz, M.E., Scalabrin, E.E.: Aligning research and product development through multi-cycle design science research: an experience report. In: *Agile Processes in Software Engineering and Extreme Programming - Workshops*, Springer Nature Switzerland, Cham (2026)
19. Winters, T.: Thoughts on applicability. *J. Syst. Softw.* **215**, 112086 (2024). <https://doi.org/10.1016/j.jss.2024.112086>
20. Woods, E.: Dear researchers step 1: find a team with a problem. *J. Syst. Softw.* **222**, 112318 (2025)
21. Yang, N., Wang, X., Zhang, Z., Siemon, D., Hyrynsalmi, S.: Core theories in agile software development. In: *Proceedings of the 26th International Conference on Agile Software Development*, pp. 3–18. Springer Nature Switzerland, Cham (2025)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





From Education to Evidence: A Collaborative Practice Research Platform for AI-Integrated Agile Development

Tobias Geger¹(✉) , Andreas Rausch¹ , Ina Schiering² , Frauke Stenzel¹ ,
and Stefan Wittek¹ 

¹ Clausthal University of Technology, Clausthal-Zellerfeld, Germany
{thomas.tobias.marcello.geger, andreas.rausch, frauke.stenzel,
stefan.wittek}@tu-clausthal.de

² Ostfalia University of Applied Sciences, Wolfenbüttel, Germany
i.schiering@ostfalia.de

Abstract. Agile software development evolves so rapidly that research struggles to remain timely and transferable—an issue heightened by the swift adoption of generative AI and agentic tools. Earlier work has highlighted a theory gap, where findings lack a clear conceptual basis for reuse, a time gap, where results arrive too late for practical decisions, and a transfer gap, where insights are not packaged in forms practitioners can readily apply. This paper introduces a project-based, AI-integrated agile education platform as a collaborative practice research environment positioned between controlled studies and real-world industry settings. The platform enables rapid inquiry through sprint rhythms, quality gates, and authentic stakeholder involvement. We present a framework that specifies iteration structures, recurring events, and quality gates for AI-assisted engineering artifacts. Early results from several semesters, including project pipeline development, cohort growth, and stakeholder participation, suggest that the platform can generate practice-relevant evidence efficiently while preserving the contextual conditions needed for reuse. Finally, we outline future steps to enhance governance and evidence capture.

Keywords: agile practice research · AI-integrated agile development · collaborative practice research platform · project-based agile education · transferability

1 Introduction

Agile software development [1] evolves rapidly, driven by distributed work, new tool ecosystems, and the rise of AI-supported development. Practice research aims to close the gap between research and practice, but major challenges remain: the theory gap, where results lack a sufficiently explicit rationale for reuse, the time gap, where industry changes faster than research cycles, and the transfer

gap, where findings are not translated into accessible forms for practitioners. Together, these gaps hinder the production of relevant, usable insights [6].

The rapid adoption of generative AI and agentic tools makes these challenges even more urgent. In this work, it is argued that AI influences not only how teams develop and test software, but also shapes core agile practices. From this perspective, this raises questions around quality, responsibility, and trust, increasing the need for timely and transferable socio-technical guidance.

Recent discussions highlight two main research strategies: research-led (Mode-1) studies using controlled, theory-driven designs, and collaborative practice research platforms (Mode-2), where researchers and practitioners co-create and test interventions in real settings, sharing insights through rapid feedback loops [6]. The latter is especially suitable for the fast-moving context of AI-integrated agile development.

We propose and show how to realize such a collaborative platform via project-based, AI-integrated agile education. Structured sprint cycles, explicit quality gates, and authentic stakeholder involvement position this approach between lab studies and industry practice, enabling rapid, contextualized research.

The study program, which will be presented in this paper, becomes a long-term collaboration space where practitioners contribute real challenges and students test solutions in sprint-based teams. This supports fast iteration, systematic evidence capture, and the packaging of transferable guidance for practice.

Our investigation is structured around three questions: (1) How can project-based, AI-integrated agile education serve as an effective collaborative research platform given these challenges? (2) How can this be implemented through a practical framework with sprints and quality gates? (3) What early results and implications emerge for timely, transferable AI-integrated agile practice?

2 Designing AI-Integrated Agile Education: AI Engineering Study Program

Teaching agile effectively requires learning formats that go beyond lecture-centric instruction: students need repeated, authentic project settings in which teamwork, evolving requirements, and continuous feedback are experienced rather than explained. At TU Clausthal, we have already operationalized this principle in the project-based study program Digital Technologies, where agile methods (e.g., Scrum), project management, creativity techniques, and teamwork are taught and internalized through practice in real projects [2], [7].

With generative AI, the need for such practice-oriented formats becomes even more pressing. LLMs and agentic AI increasingly support engineering activities, but they also introduce risks regarding quality, traceability, and responsible decision-making. Many curricula still implicitly prepare students for a workplace without AI, creating a qualification gap. We therefore follow the same core educational hypothesis that proved successful for agile education: project-based learning is a powerful vehicle to develop AI competence in context, because it

forces students to apply AI tools under real constraints, to reflect on limitations and failure modes, and to take responsibility for outcomes through human oversight.

Building on these foundations, we developed the AI Engineering study program concept. Its key idea is to combine a shared core curriculum with discipline-specific specialization tracks. The core curriculum establishes a common baseline across all students, covering programming and software foundations, AI literacy and data-related skills, and agile development practices so that students can collaborate in interdisciplinary settings. On top of this base, students choose an engineering focus (e.g., AI Software Engineering, AI Mechanical Engineering, AI Material Engineering), in which domain-specific competencies are developed and repeatedly connected to the use of AI as an engineering tool (“AI as a tool for engineering,” not only “AI as a product”).

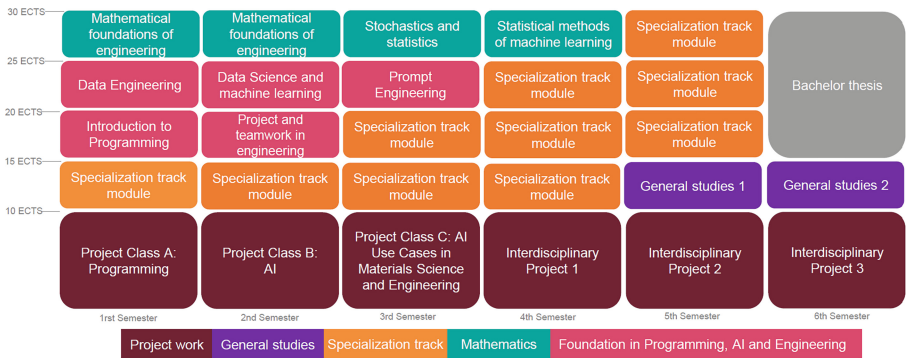


Fig. 1. Model curricula of the bachelor program AI Engineering

A distinctive structural element of the program is that a substantial share of the workload is realized in project-oriented formats (approximately one third), where students apply agile methods and AI tools end-to-end on realistic problems: iteratively, in teams, and with continuous feedback. These projects are designed as authentic socio-technical learning settings in which students experiment with AI-supported workflows, evaluate results critically, and learn how to embed AI usage into engineering quality assurance and accountability. The curriculum itself is treated as a learning, agile system that is iteratively improved with input from students, educators, industry, and societal stakeholders, and designed for transfer (e.g., reusable materials and modular extensions).

The structure of the study program with the regarding European Credit Transfer and Accumulation System (ECTS) is summarized in the model curricula shown in Fig. 1: they make explicit (i) the shared core curriculum that establishes a common foundation in programming, mathematics, AI/data competencies, and agile ways of working, and (ii) the subsequent engineering specialization tracks that deepen domain-specific knowledge while maintaining a consistent AI-augmented engineering perspective. Importantly, the figures also highlight the

program’s strong project-based backbone: project classes and recurring interdisciplinary projects are distributed across the semesters to ensure that students repeatedly apply agile practices and AI tools in authentic team settings—first to build fundamental skills, then to develop and evaluate discipline-specific AI use cases, and finally to integrate them in larger interdisciplinary projects and the bachelor thesis.

From Study Program to Collaborative Practice Research Platform. Beyond education, the program functions as a collaborative practice research platform for AI-integrated agile development. In this platform, university lecturers and researchers frame questions and synthesize insights, external stakeholders bring real problems and constraints, and students implement and evaluate sprint-based interventions under realistic conditions. Evidence is captured through a recurring artifact backbone, including backlogs and user stories, repositories, architecture and test artifacts, review presentations, retrospectives, and explicit quality gates such as review parties, repository-based assessments, and oral exams. The point is not only to observe learning, but to capture evidence of AI-supported engineering practice, decision-making, and accountability.

3 AI-Integrated Agile Project Framework: Case Study

This section reports a case study of our second-semester bachelor project in which teams conduct AI-integrated agile development, i.e., they deliberately use AI tools as part of their everyday engineering work while remaining accountable for outcomes. The following subsections operationalize our AI-Integrated Agile Project Framework through roles, iteration structures, learning mechanisms, and quality gates. Students enter with basic skills in Python, knowledge of object-oriented programming in Java, and familiarity with fundamental development tools such as Git. In addition, foundations in UML-based modeling, Scrum-based development, and project management have been established in prior courses. This project serves as a representative example of the projects conducted in later semesters, since all structural elements (Scrum framework), and time schedules, like presenting (intermediate) results, are consistent between all offered projects.

3.1 Roles and Responsibilities (Including AI Accountability)

Teams typically consist of four to six developers, a Product Owner (teaching assistant), and a Supervisor (professor). While students initially act as developers, they gradually assume Product Owner responsibilities, for instance by writing user stories and refining acceptance criteria. The technical goal of the project is to develop a browser-based multiplayer game based on a given physical board game [3]. In AI-integrated work, a key expectation is accountability: teams must be able to explain and justify AI-supported decisions and artifacts (e.g., code, tests, and design choices), including limitations and risks.

3.2 Iteration Rhythm and Events

The project follows the Scrum framework over seven two-week sprints. Standard events include bi-weekly update meetings called “Dailies”, mid-sprint “InBetween” presentations, and Sprint Reviews. Additionally, cross-team “Review Parties” are designed after sprints 3, 5, and 7 to present results to a wider audience and to create comparable checkpoints across teams.

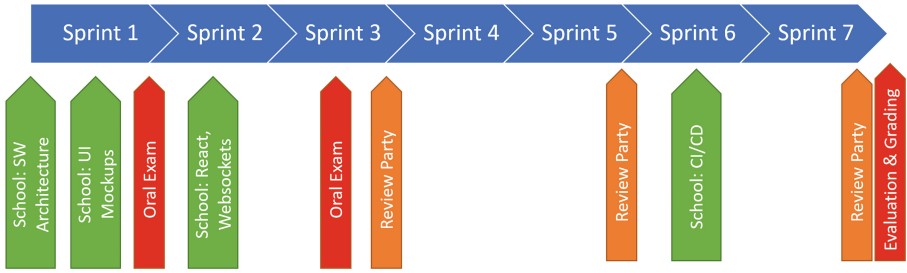


Fig. 2. Time schedule of AI-integrated Agile Projects

Figure 2 summarizes the time schedule of a typical project, including the integration of different learning units (Schools) into the sprint cadence.

3.3 Learning and Transfer Mechanisms

To introduce new knowledge, we employ “Schools”—hybrid lecture-exercise units on technology and architecture. AI assistance is embedded into each School; instructors demonstrate capabilities and limitations (e.g., refactoring, testing) alongside technical concepts. Assignments require students to verify AI outputs against project constraints. This approach supports learning by providing authentic AI usage under pressure and improves transfer by making AI usage a visible, repeatable process. Importantly, AI use is not rewarded as tool use itself, but only when students can justify its contribution to an accountable, correct solution.

3.4 Quality Gates and Evidence Points

While AI is used throughout, we ensure deep understanding through explicit quality gates: individual oral exams (20 min) and presentations serving as living documentation. There, the team members have to convince with their knowledge. This creates a lightweight traceability layer that prevents superficial AI use and serves as a recurring evidence point for research, ensuring that AI-supported engineering decisions are understandable and accountable. In practice, the relevant research evidence is the combination of AI inputs, human judgment, and the resulting project artifacts.

4 Early Findings: Observations and Implications

This section reports early evidence from applying the proposed collaborative practice research platform in the Digital Technologies/AI Engineering context. Our goal is not to claim generalizable effects, but to surface recurring observations that (i) indicate the feasibility of the platform approach at scale, (ii) highlight where it helps to mitigate timeliness and transfer challenges in AI-integrated agile work, and (iii) form implications for future research and platform refinement.

4.1 Evidence Sources and Scope

We triangulate evidence across (a) project pipeline statistics, (b) cohort indicators, and (c) recurring artifacts (e.g., review presentations, repositories, oral exams). Our analysis draws on these sources in a complementary way: pipeline statistics indicate scale, artifacts show project outcomes, and reflection points reveal decision-making processes. Qualitative signals were identified by comparing patterns across teams and semesters, paying attention to confirming and disconfirming cases to separate isolated incidents from recurring practice patterns. This approach allows us to derive implications for platform development while transparently acknowledging the limits of this work-in-progress stage.

4.2 Quantitative Snapshot: Project Pipeline, Stakeholder Involvement, and Growth

Across the observed semesters, the platform sustained a stable project throughput while cohort sizes increased. In the period from summer semester 2023 to winter semester 2025, the number of conducted projects grew from 7 to 18 per semester, with typical team sizes ranging from 4 to 6 students. Over the same period, the number of students participating in projects increased from 26 to 111. The project pipeline is intentionally over-provisioned: for example, in winter semester 2025, 20 projects were pitched and 18 were conducted, enabling a selection mechanism aligned with student interest while maintaining sufficient capacity.

A key indicator for transfer-oriented, AI-integrated agile work is authentic stakeholder involvement. Industry (real-stakeholder) projects were continuously offered each semester, typically in the range of 1-3 projects, and covered diverse partners and domains (e.g., mobility, smart city, and AI-enabled systems). This recurring stakeholder participation provides a stable channel for bringing in current constraints, expectations, and success criteria, while keeping project themes connected to fast-moving practice needs.

The platform also evolved under clear growth pressure. Overall program size increased substantially over time, especially on the master's level, resulting in larger and more diverse cohorts. Such growth is important for practice research: it increases the opportunity for repetition and cross-team triangulation, and it creates a broader empirical base for identifying recurring patterns rather than one-off anecdotes.

4.3 Observations and Implications

O1: A Sprint-Based Project Platform can Scale While Preserving Repeatability. Even with more students and projects, the platform maintained a regular process, enabling rapid, comparable research cycles. *Implication:* Favor regular sprint rhythms and checkpoints over ad-hoc structures for faster feedback and comparability.

O2: Over-provisioning (Pitched vs. Conducted Projects) acts as a Built-in Selection and Relevance Mechanism. More projects are pitched than conducted; only popular or relevant ones proceed, supporting timely validation of emerging topics. *Implication:* Use project selection as a relevance signal to keep research current.

O3: Continuous Real-Stakeholder Participation Strengthens Transfer by Making Context Explicit from the Outset. Industry involvement gives early clarity on constraints and criteria, which helps make findings reusable. *Implication:* Institutionalize stakeholder involvement as an essential part of the platform.

O4: Quality Gates Create Structured Evidence Points for AI-integrated Work (Beyond Tool Use). Quality gates require teams to clearly justify AI-driven decisions, focusing on accountability over mere tool use. *Implication:* Design quality gates for traceability, explainability, and responsibility.

O5: The Recurring Artifact Set Enables Packaging for Reuse and Cumulative Learning. Repeated semester artifacts, such as backlog items, review slides, repositories, and retrospective notes, make it possible to compare teams and extract transferable patterns. *Implication:* Standardize a minimal artifact structure and extend it with AI-specific traceability elements, such as prompt logs or decision notes, when needed.

O6: Growth Increases Triangulation Potential, but also Demands Stronger Platform Governance. Larger cohorts enable more cross-team insights, but also require better coordination and standards. *Implication:* Apply explicit governance to maintain quality and comparability when scaling.

4.4 Summary

Overall, the early evidence suggests that project-based AI-integrated agile education can function as a collaborative practice research platform that supports rapid cycles of inquiry (timeliness) while improving transfer through stakeholder anchoring and systematic artifact packaging. At the same time, the observations highlight that scale and transfer do not happen automatically: they depend on explicit platform governance, consistent quality gates, and deliberate documentation practices tailored to AI-integrated engineering work.

5 Discussion and Conclusion

Limitations of this study include its strong reliance on qualitative methods such as observations during review parties, retrospectives, and oral exams. Future work should complement these with quantitative measurements [4] and further examine how agile mindset and culture development [5] can be supported through respect, trust, learning, and feedback culture.

This case study presents a collaborative practice research platform (Mode-2) for co-creating and testing interventions in real-world settings with rapid feedback loops [6]. Key scaling challenges include maintaining project relevance, authentic stakeholder integration, and effective platform governance to ensure quality and comparability. Future research will therefore focus on quantitative evaluation and stronger stakeholder participation from both research and practice.

A key transfer value of the platform is its ability to generate reusable, practice-oriented outputs beyond isolated case studies. Standardized artifacts across semesters, such as documentation templates, AI-usage checklists, prompt-based decision logs, and review rubrics, help translate empirical findings into adaptable tools for practitioners.

In conclusion, project-based, AI-integrated agile education can serve as an effective collaborative practice research platform that improves research timeliness and bridges the transfer gap. Future work will focus on validating and packaging recurring patterns into practitioner-ready protocols .

Acknowledgments. This study was funded by the Lower Saxony Ministry for Science and Culture (grant number 11 – 76251-1900/2021 and 11 – 76251-1899/2021).

References

1. Beck, K., et al.: Agile manifesto. softw dev (2001)
2. Center for Digital Technologies: Digital Technologies (2026). <https://digitecstudieren.de>. Accessed 20 Jan 2026
3. Center for Digital Technologies: Zweitsemesterprojekte (2026). <https://projects.digitecstudieren.de>. Accessed 20 Jan 2026
4. Choraś, M., et al.: Measuring and improving agile processes in a small-size software development company. *IEEE Access* **8**, 78452–78466 (2020). <https://doi.org/10.1109/ACCESS.2020.2990117>
5. Neumann, M., Kuchel, T., Diebold, P., Schön, E.M.: Agile culture clash: unveiling challenges in cultivating an agile mindset in organizations (2024). <https://arxiv.org/abs/2405.15066>

6. Neumann, M., Schön, E.M., Senapathi, M., Rauschenberger, M., da Silva, T.S.: Towards a closer collaboration between practice and research in agile software development workshop: a summary and research agenda (2025). <https://arxiv.org/abs/2507.10785>
7. Stenzel, F., Küpper, S., Rausch, A., Schiering, I., Bikker, G.: Projektbasierte Lehre in der Informatik zur Entwicklung von Zukunftskompetenzen. In: Dölling, H., Schäfle, C. (eds.) Tagungsband zum 6. Mint Symposium: Zukunft MINT Lehre: Was bleibt? Was kommt? Was wirkt? pp. 170 – 177. BayZiel, München (2025). https://doi.org/10.57825/repo_in-6376

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Aligning Research and Product Development Through Multi-Cycle Design Science Research: An Experience Report

Alexandre Nodari¹(✉) , Marcelo Luis Walter¹ , Leonardo Nunes² ,
Marcelo E. Pellenz¹ , and Edson Emilio Scalabrin¹

¹ Pontifícia Universidade Católica do Paraná, Curitiba, Brazil
alenodari@gmail.com , scalabrin@ppgia.pucpr.br

² Sumersoft Tecnologia Ltda, Curitiba, Brazil
leonardo@sumersoft.com

Abstract. University-industry collaborations in software engineering are critical for innovation but often fail due to misaligned incentives and timelines. This experience report presents a successful partnership in developing an AIoT product that addresses this gap. We introduce and evaluate the Multi-Cycle Design Science Research (DSR) framework, an approach designed to synchronize research and product development. The framework structures the collaboration into fixed-length cycles, in which the primary deliverable of each cycle is a peer-reviewed scientific publication that simultaneously addresses a critical product development challenge. Over four cycles, this method contributed to progressively de-risking the project, guiding it from concept toward a commercially viable product, while producing four academic articles. Our contribution is a practical model for structuring industry-academia collaborations that explicitly aligns the goals of both partners by treating academic publications as first-class business deliverables. This approach offers a tangible solution to bridge the research-practice gap in agile and technology-driven projects.

Keywords: Design Science Research · University-Industry Collaboration · Agile Development · Research-Practice Gap · Experience Report

1 Introduction

University-industry collaborations are recognized as powerful engines for innovation, combining the exploratory, deep knowledge of academia with the market-driven, practical focus of industry [9]. However, these partnerships are difficult

This work has been supported by “Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) Código de Financiamento 001.”

© The Author(s) 2027

A. Goldman et al. (Eds.): XP 2026 Workshops, LNBIP 596, pp. 22–30, 2027.

https://doi.org/10.1007/978-3-032-39205-3_3

to sustain. Many R&D initiatives fail to transition from research prototypes to commercially viable products [4]. The root causes are often traced back to a misalignment of goals, timelines, and cultures. Academia typically operates on longer cycles, driven by the pursuit of novel research and the need to publish peer-reviewed articles. At the same time, industry demands rapid, tangible results and a clear return on investment [7, 8].

Agile methodologies, with their emphasis on iterative development, short feedback loops, and close collaboration, have been proposed as a potential solution to bridge this divide [1]. By breaking down large, complex projects into manageable increments, they can introduce a rhythm and transparency that traditional, waterfall-style R&D management lacks. However, the simple adoption of agile ceremonies is often insufficient to resolve the core incentive conflict. A cycle that delivers a software feature may be a victory for the industry partner. Still, it does not directly contribute to the primary currency of an academic researcher: a scientific publication.

This experience report presents a successful R&D collaboration that directly addresses this challenge. The project, focused on developing an innovative device for smart metering, was a partnership between a software company and our university. A key strategic decision in forming this partnership was the selection of a research team composed of graduate students who were also well-versed in agile principles. This deliberate choice mitigated many of the typical cultural and communication barriers from the outset, creating a foundation of shared language and values [10].

Building on this foundation, we implemented a novel agile framework to explicitly align the incentives of both partners. We structured the collaboration around a series of **research cycles based on Multi-Cycle Design Science Research**, in which the primary macro-deliverable of each cycle was a scientific publication that simultaneously addressed a critical product development gap. Instead of treating academic and industrial outputs as separate objectives, this approach fused them into a single, unified goal. The research question for each publication was the core business problem, and the resulting peer-reviewed paper served as a validated solution and a project milestone.

This report documents how this framework was applied over four consecutive cycles to de-risk and develop the product, moving from initial technical feasibility to machine learning model optimization, hardware cost-benefit analysis, and finally to state-of-the-art contextualization through systematic literature review. We report on our experience with this symbiosis of goals, which not only fostered a robust partnership but also supported product development in a technology-driven domain. Our contribution is a practical, replicable model for structuring agile collaborations between academia and industry, addressing a long-standing problem and providing a detailed account relevant to Agile ML Engineering and industry-academia partnerships.

1.1 Related Work

University-industry collaborations (UIC) are essential for driving innovation in software engineering (SE), yet they are fraught with challenges. A primary challenge is the misalignment of goals and timelines between academic pursuit of knowledge and industrial demand for immediate, tangible results [2, 5].

Garousi et al. [2] provided a broad characterization of UICs in SE by surveying 101 projects. Their findings highlight that while over 90% of collaborations result in publications, the primary drivers and success factors vary. They identified common patterns, such as the need for regular communication and clear goals, and anti-patterns, such as a lack of trust.

In contrast, Marijan and Gotlieb [5] proposed the Certus model, a more structured, phase-based approach for fostering participative knowledge co-creation. Developed from an 8-year longitudinal study, the model outlines seven phases, from project setup to the exploitation of results, emphasizing continuous dialogue and joint problem-solving.

Our Multi-Cycle DSR framework contributes to this discourse by offering a more granular and operational model. It introduces a key mechanism for operationalizing the collaboration: fixed-length cycles where the primary deliverable is a peer-reviewed scientific publication. This approach elevates publications to a first-class deliverable that serves as both a validation artefact and a driver of product development, thereby creating a tangible and powerful alignment of incentives.

2 The Smart-Eyes Project for Utility Metering

The subject of this experience report is the Smart-Eyes project, a commercial endeavor by the software company Sumersoft. The product is an innovative, retrofittable AIoT (Artificial Intelligence of Things) device designed to convert conventional analog water and gas meters into smart meters. Its core value proposition is to provide property managers, construction companies, and condominium associations with the benefits of smart metering—such as remote reading, leak detection, and real-time consumption monitoring—without the prohibitive cost and complexity of replacing existing infrastructure.

The Smart-Eyes device is a compact, battery-powered unit that is physically attached to the face of an analog meter. It uses a low-resolution camera to periodically capture images of the meter’s mechanical dial. These images are then processed locally by an onboard machine learning model, which performs optical character recognition (OCR) to read the digits. The extracted consumption data is then transmitted wirelessly via a LoRaWAN network to a central server, making the information accessible through a web and mobile platform.

2.1 The Initial Challenge and Partnership

The project’s primary technical hurdle was the significant challenge of running a sufficiently accurate machine learning model on a resource-constrained edge

device. The chosen hardware platform, the ESP32-S3 microcontroller, offered a compelling balance of low cost and energy efficiency, but it possessed limited processing power and memory. Initial in-house attempts by the company to develop an OCR model were unsuccessful in achieving the required accuracy for a commercial product. The models were either too large and computationally expensive to run on the microcontroller or too simplistic to reliably read the digits under the variable real-world conditions of lighting, angle, and meter degradation. This created a critical business risk: without a viable edge ML model, the entire product concept was technically unfeasible.

To address this challenge, Sumersoft initiated a formal partnership with the Pontifical Catholic University of Paraná (PUCPR), a university with a strong graduate program in computer science and applied informatics. The collaboration was structured around the allocation of graduate-level researchers (Master’s and Ph.D. students) to the project. A strategic element of this partnership was the recruitment of researchers who possessed a dual background: strong academic credentials combined with significant prior experience as software engineering professionals familiar with agile methodologies. This intentional selection created a team well-positioned to understand both the rigor of scientific research and the pragmatic demands of product development.

3 Aligning Research and Product Goals

To overcome the typical friction points of university-industry collaborations, we designed and implemented a bespoke agile framework grounded in a core principle: the explicit alignment of academic and industrial incentives. Instead of viewing research and product development as parallel, often conflicting tracks, we fused them into a single, symbiotic process. The cornerstone of this framework was the establishment of fixed-length, high-level research cycles based on **Multi-Cycle Design Science Research**.

Our approach is an instantiation of the Design Science Research (DSR) methodology, which provides a canonical process model for creating and evaluating artifacts that solve organizational problems [6]. The methodological novelty of our **Multi-Cycle DSR** lies in its adaptation for agile, industry-driven projects. We decompose a large, complex product development challenge into a series of smaller, logically coherent, and self-contained research cycles. Each cycle functions as a complete DSR project in itself, executing all of Peffers et al.’s phases (Problem Identification, Objective Definition, Design and Development, Demonstration, Evaluation, and Communication), and is inspired by similar DSR applications in resource-constrained environments [3]. The key distinction is at the *Communication* phase with the submission of a peer-reviewed scientific publication, which in turn serves as the primary *Evaluation* artifact for the business.

3.1 The Research Cycle Structure and Workflow

The framework’s cadence was built around nine-month cycles. This specific duration was chosen as a strategic compromise: it was short enough to provide the rapid feedback and tangible progress demanded by the industry partner, yet long enough to allow for the depth of research, experimentation, and writing required to produce a high-quality scientific publication for the academic partner.

The defining feature of each cycle was its goal: **to produce a peer-reviewed scientific paper that directly answered a critical, unresolved question hindering the product’s development.** This transformed the academic publication from a secondary, often delayed, byproduct of the collaboration into the primary, value-driven milestone for both parties.

The workflow of each cycle followed a clear, repeatable pattern. The process began with the industry partner identifying a major product gap or risk. This business problem was then reframed as a formal research question by the academic team. The subsequent research and development work was a collaborative effort to solve this question, with the findings being continuously integrated into the product’s codebase and simultaneously documented for the scientific paper. The culmination of the cycle was the submission of the paper to a relevant conference or journal. This event served as a hard deadline and a clear “definition of done” for the cycle. The outcome was twofold: the company received a validated solution to a critical problem, de-risking its product roadmap, while the researchers achieved a tangible academic output, advancing their studies and careers.

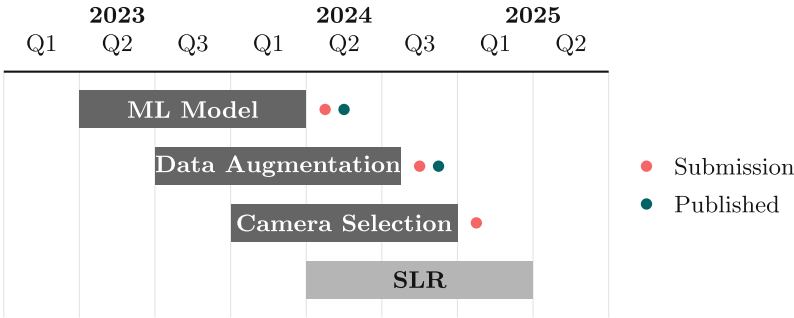


Fig. 1. Gantt Chart of cycles (2023–2025).

Figure 1 illustrates the temporal distribution of the research cycles, showing how they were structured over time with publication milestones and demonstrating the concrete application of the Multi-Cycle DSR approach.

4 Lessons Learned: The Four Cycles

Cycle 1: ML Model Selection. The first cycle addressed the core question of technical feasibility. This business problem was translated into our first research

question (RQ1): *Which machine learning model architecture provides the best trade-off between digit recognition accuracy and resource consumption for deployment on an ESP32-S3 microcontroller?* The research team focused on exploring and evaluating various lightweight machine learning architectures suitable for edge deployment. The work involved developing a proof-of-concept, training initial models, and rigorously testing their performance in a controlled environment that simulated the device’s operational constraints. This cycle culminated in the first scientific publication [11], which demonstrated that it was feasible to achieve promising accuracy in digit recognition on the target hardware while maintaining extremely low power consumption. The successful completion of this cycle provided an initial green light for the project to proceed, reducing the perceived risk of the core technology and giving the industry partner the confidence to continue investing in the collaboration.

Cycle 2: Data Augmentation. While the first cycle proved feasibility, the baseline accuracy of the initial model was not yet robust enough for a commercial-grade product. This led to our second research question (RQ2): *To what extent can data augmentation techniques compensate for the lack of a large, real-world dataset in training a robust OCR model for analog meters?* The team conducted a systematic investigation into various augmentation methods, such as brightness adjustments, translations, and elastic deformations, applying them to both public and the project’s private datasets. This research led to the second publication [12], which identified effective augmentation strategies and a saturation point where further increases in volume offered diminishing returns. The application of these findings contributed to a meaningful improvement in the model’s accuracy bringing it closer to the level required for commercial deployment.

Cycle 3: Camera Selection. With the software and ML model proven, the focus shifted to hardware optimization. The need to balance cost and performance was framed as our third research question (RQ3): *What is the impact of camera module cost and resolution on the performance of an edge-based OCR model, and what is the optimal cost-benefit trade-off?* The camera module was a significant component of the device’s bill of materials, and reducing its cost was essential for improving the product’s profitability and market competitiveness. The third cycle was structured as an engineering-focused investigation into the cost-benefit trade-offs of various low-cost camera modules. The team established a rigorous, evidence-based methodology to evaluate a selection of cameras, using objective metrics for image quality and operational performance. This research enabled the company to make an informed, evidence-based decision on hardware selection, resulting in the third publication, choosing a camera that contributed to reducing the unit cost while aiming to maintain the performance required by the ML model.

Cycle 4: Systematic Literature Review. With the core technical challenges resolved and the product approaching commercial viability, the research needed to be positioned within the broader academic landscape. This need for academic

positioning defined our final research question (RQ4): *What is the current state of the art in Edge AI for smart utility metering, and what are the principal open research gaps?* The fourth cycle was dedicated to conducting a Systematic Literature Review (SLR) on the application of Edge AI in smart metering systems. The team followed a rigorous protocol to search, filter, and analyze relevant academic publications, mapping the predominant architectures, techniques, challenges, and trends in the field. This research confirmed that the approach adopted in the previous three cycles was aligned with current best practices, particularly regarding the use of CNNs on ESP32-S3 microcontrollers for meter reading. More importantly, the review revealed a significant gap in the literature: the integration of time-series-based anomaly detection directly on edge devices and the development of Digital Twins for smart metering with leak detection capabilities. This positioned the project's future work on Digital Twins and anomaly detection as an original and necessary contribution to the field, as documented in the fourth publication.

5 Discussion and Limitations

While our experience with the Multi-Cycle DSR framework was positive, its success was contingent on specific boundary conditions. These include: researchers with dual industry-academic backgrounds, strong executive sponsorship from the industry partner, and a problem domain with clear, researchable challenges divisible into self-contained, publishable units. The framework may be less applicable in contexts lacking these conditions. Additionally, tying research to product milestones carries the risk of using scientific methods to justify pre-existing business decisions rather than fostering open inquiry. We mitigated this by framing each business problem as a formal, open-ended research question *before* development began. Another operational risk is the dependency on peer-review timelines, where a rejection could disrupt the nine-month cadence.

Finally, while a formal evaluation of the collaborative process was outside the scope of this report, informal feedback from both academic and industry participants was consistently positive, highlighting the clear alignment of goals and the predictable rhythm of deliverables as key factors in reducing friction.

6 Conclusion

This experience report presented our university-industry collaboration for the development of an innovative AIoT product. We addressed the persistent challenge of aligning academic and industrial goals by implementing a novel agile framework based on **Multi-Cycle Design Science Research**. Our approach structured the collaboration around a series of research cycles, where the primary deliverable was a scientific publication that simultaneously solved a critical product development gap.

The results from the Smart-Eyes project demonstrate the effectiveness of this model. Over four consecutive cycles, this symbiotic approach progressively

de-risked the project, transforming it from a technically uncertain concept into a product approaching commercial viability with a validated research roadmap. The framework provided a steady rhythm of tangible progress, fostered trust, and, most importantly, created a powerful alignment of incentives between the academic and industrial partners.

Our primary contribution is a practical and replicable model for structuring agile collaborations in complex, technology-driven R&D projects. By treating peer-reviewed publications not as a secondary academic pursuit but as a first-class business milestone, we turned a potential source of conflict into an engine for innovation. This experience report suggests that it is possible to create an agile symbiosis between academia and industry, where the rigor of science drives the quality of the product, and the needs of the market drive the relevance of the research.

References

1. Fernandes, G., et al.: Project management practices for collaborative university-industry r&d: a hybrid approach. *Procedia Computer Science*. **138**, 717–725 (2018)
2. Garousi, V., Pfahl, D., Fernandes, J.M., Felderer, M., Mäntylä, M.V., Shepherd, D., Arcuri, A., Coşkunçay, A., Tekinerdogan, B.: Characterizing industry-academia collaborations in software engineering: evidence from 101 projects. *Empir. Softw. Eng.* **24**, 2540–2602 (2019)
3. Herselman, M., Botha, A.: Evaluating an artifact in design science research methodology as was implemented in a resource constrained environment (2015). <https://doi.org/10.1145/2815782.2815806>
4. Johnson, W.C.: Challenges in university-industry collaborations. In: *Universities and Business: Partnering for the Future*. Sense Publishers (2006)
5. Marijan, D., Gotlieb, A.: Industry-academia research collaboration in software engineering: the Certus model. *Inf. Softw. Technol.* **132**, 106473 (2021)
6. Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A design science research methodology for information systems research. *J. Manage. Inf. Syst.* **24**(3), 45–77 (2007). <https://doi.org/10.2753/MIS0742-1222240302>
7. Rossoni, A.L.: Barriers and facilitators of university-industry collaboration. *PMC* (2023)
8. Schofield, T.: Critical success factors for knowledge transfer between university and industry. *J. Res. Adm.* **44**(2), 38–56 (2013)
9. Varma, B.M.: Exploring effective collaboration models in industry-academia partnerships. In: *International Conference on Work Integrated Learning* (2024)
10. Vuoriainen, A.: The six c's of successful higher education-industry collaborations. *Industry and Higher Education* (2025)
11. Walter, M.L., et al.: Evaluation of machine learning models in a smart water metering system, Kuching, Malaysia, pp. 2651–2656. *IEEE* (2024). <https://doi.org/10.1109/SMC54092.2024.10831187>
12. Walter, M.L., et al.: Fine-tuned data augmentation techniques for digit recognition in AIoT. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) *Artificial Intelligence and Soft Computing*, pp. 209–221. Springer, Cham (2026). doi: https://doi.org/10.1007/978-3-032-03708-4_17

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Designing Transfer-Ready Research Artefacts for Ways of Working

Philipp Diebold^{1,2} 

¹ Bagilstein GmbH, Mainz, Germany

philipp.diebold@bagilstein.de, philipp.diebold@iu.org

² IU International University, Erfurt, Germany

Abstract. Despite decades of research on Ways of Working, a persistent gap remains between academic insights and their adoption in practice. While prior work has extensively discussed the theory-practice gap, less attention has been paid to the design of transfer itself. This paper argues that transfer is not an automatic by-product of research dissemination, but a deliberate design activity requiring its own principles, artefacts, and validation criteria.

Using a canvas as an illustrative case, this paper reflects on how research-based concepts on ways of working can be translated into a form that is usable in practice without losing theoretical grounding. The canvas integrates insights from empirical research on context-dependent working practices and human behavior into a lightweight diagnostic artefact that can be applied by practitioners without prior exposure to the underlying theories.

The paper contributes a structured reflection on how the theory-gap, time-gap, and transfer-gap manifest in research on ways of working, an analysis of concrete design decisions that enabled research-to-practice translation in the canvas, and a set of transfer-oriented design principles that may guide future research artefacts. The work aims to support more impactful collaboration between academia and industry by reframing transfer as a core research concern rather than an afterthought. Rather than claiming generalizable evidence of effectiveness, the paper provides an exploratory and design-oriented reflection on transfer-readiness as a quality criterion for research artefacts.

Keywords: ways of working · research-to-practice transfer · boundary objects · design principles · diagnostic artefacts

1 Introduction

Ways of working have been the subject of extensive academic investigation for many years, spanning topics such as collaboration models, coordination mechanisms, leadership styles, process structures, and organizational contexts. Despite this growing body of knowledge, practitioners frequently report limited benefit

from academic research when addressing concrete challenges in their day-to-day work.

This disconnect is often described as a theory-practice gap. However, framing the problem solely as a lack of practical relevance or insufficient rigor risks oversimplification. In many cases, relevant research does exist, but it fails to reach practice in a form that is usable under real-world constraints such as time pressure, cognitive load, and organizational complexity.

Recent discussions have therefore expanded the perspective to include additional dimensions, such as the time-gap - the mismatch between academic research cycles and the pace of organizational change - and the transfer-gap, referring to the difficulties of translating research insights into actionable forms for practitioners.

We believe that the transfer-gap deserves particular attention: Rather than treating transfer as the final step of a research project (e.g., via publications or presentations) [1], transfer should be understood as a design problem: How must research outcomes be shaped so that they can be meaningfully applied in practice?

To explore this question, the paper uses the Genau Meine Arbeitsweise¹ (GMA) Canvas as an illustrative case. The GMA Canvas was developed based on empirical research into ways of working and contextual decision-making [7], but deliberately designed as a lightweight, practitioner-facing artefact. By reflecting on its development and use, this paper seeks to derive more general insights into the design of transfer-ready research artefacts.

In addition, research on ways of working increasingly acknowledges that no single model or framework can be universally applied across organizational contexts. Contextual factors such as organizational culture, regulatory environments, team maturity, and individual motivation strongly influence whether a particular way of working is effective or even feasible [4].

Despite this awareness, many research outputs still implicitly assume that practitioners are able and willing to translate abstract findings into their specific context. This assumption often underestimates the translation effort required and overlooks the fact that practitioners typically operate under severe time constraints and competing priorities.

Consequently, the question is not only whether research findings are relevant, but whether they are *transfer-ready*: shaped in a way that allows meaningful application without requiring deep methodological or theoretical expertise. This paper positions transfer-readiness as a key quality criterion for research artefacts addressing ways of working. The work follows an exploratory and reflective design-oriented perspective inspired by Design Science Research (DSR), which emphasizes the creation and refinement of artefacts addressing practical problems [10]. Rather than aiming to provide a formal evaluation of effectiveness, the paper focuses on reflecting design decisions, transfer considerations, and observations gathered during iterative use of the artefact in practice-oriented settings.

¹ English Translation would be “My Way of Working”.

2 Background: Gaps Between Research and Practice in Ways of Working

2.1 The Theory-Gap

The theory-gap refers to the misalignment between academic theories on ways of working and the needs of practitioners. Research often produces abstract models, constructs, or taxonomies that aim for generalizability and explanatory power. Practitioners, by contrast, tend to seek orientation, decision support, and context-sensitive guidance rather than comprehensive theoretical explanations.

As a result, even theoretically sound models may remain unused if they require significant interpretation effort or methodological expertise. In practice, theory frequently becomes visible only indirectly, for example through simplified heuristics or generalized best-practice narratives.

This gap is further reinforced by differing success criteria in academia and practice. Academic contributions are primarily evaluated based on rigor, novelty, and theoretical contribution (such as number of papers, citations, etc.), whereas practitioners assess usefulness based on clarity, applicability, and immediate value [13].

As a result, even well-founded theories on ways of working may struggle to gain attraction if their practical implications are not explicitly articulated or embedded into usable formats. In many cases, practitioners rely instead on simplified interpretations propagated through consulting practices, practitioner books, or informal community knowledge, which may dilute original research insights.

2.2 The Time-Gap

The time-gap describes the mismatch between academic research cycles and the tempo of organizational reality. Rigorous empirical studies, peer review, and publication processes are inherently time-consuming. Meanwhile, organizations face immediate challenges and must continuously adapt their ways of working.

This gap is particularly stated in dynamic environments, where working practices evolve rapidly in response to technological, organizational, and social change. [2] By the time research findings are published, practitioners may already have adjusted their approaches or moved on to new interpretations.

Moreover, the increasing pace of organizational change amplifies this challenge. Digital transformation, remote collaboration, and evolving workforce expectations continuously reshape ways of working, often faster than traditional research cycles can respond [2].

This temporal mismatch can lead practitioners to perceive academic research as outdated or disconnected from current realities, even when the underlying principles remain valid. Addressing the time-gap therefore requires not only faster dissemination mechanisms, but also research outputs that are resilient to change and adaptable over time.

2.3 The Transfer-Gap

The transfer-gap goes beyond relevance and timeliness. It concerns the form in which research is communicated and embedded into practice. Even when research is relevant and timely, it may fail to create impact if it is not translated into artefacts that align with practitioners' everyday ways of working.

Common transfer mechanisms - such as academic papers, guidelines, or maturity models - often assume a level of motivation, time, and methodological competence that does not reflect daily practitioners practice. Consequently, the burden of translation is placed on practitioners, who must interpret and adapt research findings themselves [15].

2.4 Interdependencies Between the Gaps

While the theory-gap, time-gap, and transfer-gap are analyzed separately, they are closely interrelated in practice. A strong theory-gap often increases the transfer-gap, as abstract models are harder to translate into actionable artefacts. Similarly, the time-gap can intensify transfer challenges when research outputs no longer align with practitioners' current concerns.

Understanding these gaps as an interconnected system highlights the need for holistic approaches that address not only the content of research, but also its form, timing, and mode of engagement with practice. This perspective motivates the focus on transfer-oriented design explored in the remainder of this paper.

3 Transfer as a Design Problem

This paper takes the position that transfer should be treated as an explicit design activity. Designing for transfer involves making conscious decisions about what to include, what to omit, and how to structure knowledge so that it can be used under practical constraints.

From this perspective, successful transfer artefacts share several characteristics:

- They reduce complexity without denying it.
- They foreground diagnosis over prescription.
- They allow use without requiring explanation of underlying theory.
- They support reflection rather than enforcing compliance.

Designing for transfer requires acknowledging that artefacts mediate between different communities. Researchers and practitioners operate with different languages, assumptions, and evaluation criteria. Transfer artefacts therefore need to function as translation devices that preserve core theoretical insights while adapting their representation to practical use [9].

From a design perspective, this implies trade-offs. Increasing usability often requires abstraction, simplification, or omission of detail. However, excessive simplification risks losing explanatory power or misrepresenting research findings.

Transfer-oriented design must therefore balance fidelity to theory with practical constraints.

In this sense, transfer artefacts are not neutral containers of knowledge, but actively shape how research is interpreted and applied. Making these design decisions explicit can improve transparency and allow more systematic evaluation of transfer effectiveness [11, 12].

Transfer-Ready Research Artefacts

Transfer-ready artefacts are research-informed instruments that are designed for practical use under real-world constraints. They embed theory implicitly, support contextual diagnosis, and can be applied without prior methodological training.

The following GMA Canvas was intentionally designed along these lines. Rather than presenting a new framework for ways of working, it provides a structured space for reflecting on context-specific working practices.

4 The GMA Canvas as a Transfer Artefact

4.1 Research Foundations

The GMA Canvas builds on empirical research into ways of working [7] [8] that emphasizes context-dependency and human decision-making. Central to its conceptual foundation is the idea that appropriate working practices emerge from the interaction of motivational factors (*willingness*), capability-related factors (*ability*), and contextual constraints (*permission*) [5] (Fig. 1).

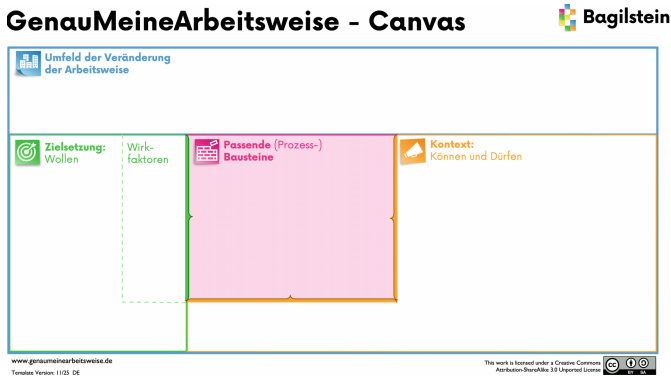


Fig. 1. Empty GMA Canvas

These dimensions are well-established in organizational and behavioral research. However, in their original form, they are typically discussed in academic language and embedded in broader theoretical frameworks.

The canvas structures reflection around these dimensions and guides teams through questions regarding current ways of working, perceived obstacles, and possible improvement directions. In practice, teams collaboratively document observations, tensions, and ideas directly within the canvas structure to create a shared understanding of their current situation.

For example, a team struggling with unclear decision-making responsibilities may identify high willingness to improve collaboration, but insufficient permission to change established processes without management approval. The canvas thereby helps to differentiate whether a problem is rooted in motivation, capability, or contextual constraints.

Importantly, these dimensions are not treated as variables to be optimized independently. Instead, the GMA Canvas emphasizes their interaction. For example, high willingness without sufficient permission may lead to frustration, while strong formal permission without corresponding ability may result in superficial adoption.

By embedding these relationships implicitly into the canvas structure, the GMA Canvas supports nuanced reflection without requiring users to explicitly reason about the underlying theoretical constructs.

4.2 Design Decisions for Transfer

Several design decisions were critical in translating these research insights into the GMA Canvas:

From Model to Canvas. Instead of presenting a formal model, the insights were embedded in a visual canvas format. This allows practitioners to engage with the concepts collaboratively and iteratively.

From Explanation to Questions. The canvas is structured around guiding questions rather than statements or rules. This shifts the focus from applying predefined solutions ("consulting") to exploring the specific context (more similar to "coaching") [14].

From Best Practices to Diagnosis. The GMA Canvas does not prescribe specific ways of working. Instead, it supports diagnosing the current working situation and identifying constraints and potentials.

From Terminology to Usability. Academic terminology is minimized. Practitioners can use the canvas effectively without knowing the underlying theories or their origins [3].

These design decisions were informed by repeated application and refinement in practice-oriented settings. Feedback from workshops and coaching sessions was used to iteratively adapt wording, structure, and visual layout, illustrating a co-evolution of artefact and context [6].

This iterative approach highlights that transfer-ready artefacts are not static outputs, but evolving instruments that must be continuously aligned with both research insights and practical realities.

4.3 Use in Practice

In practice, the GMA Canvas has been used in workshops, coaching sessions, and team reflections across different organizations as well as roles. Its value lies less in producing definitive answers than in enabling structured thinking about existing and desired ways of working and documenting this.

This illustrates how transfer-ready artefacts can act as boundary objects between research and practice, facilitating shared understanding without requiring full alignment on theoretical assumptions.

4.4 Limitations and Scope

While the GMA Canvas illustrates one approach to designing transfer-ready research artefacts, it does not claim universal applicability. Its effectiveness depends on facilitation quality, organizational openness, and the willingness of participants to engage in reflective dialogue.

In addition, the presented reflections are closely connected to the author's dual role as both researcher and practitioner involved in the artefact's design and application. This creates potential risks of confirmation bias and limits independent validation of the observations presented in this paper.

Furthermore, this paper focuses on reflective and exploratory insights rather than formal empirical evaluation of effectiveness. The contribution should therefore be understood as an initial design-oriented perspective on transfer-readiness rather than a validated transfer model. Future studies may complement this perspective with systematic empirical assessments across different contexts and scales.

5 Implications for Research on Ways of Working

Reflecting on the GMA Canvas case suggests several implications for research on ways of working:

- **Transfer requires its own design criteria.** Research quality alone does not guarantee practical impact.
- **Artefacts can carry theory implicitly.** Making theory “invisible” at the point of use may increase adoption.
- **Speed and iteration matter.** Transfer artefacts should be adaptable and evolve alongside practice.
- **Collaboration is essential.** Designing for transfer benefits from close interaction between researchers and practitioners.

Beyond individual artefact design, the findings suggest broader implications for research on ways of working. Transfer considerations should be integrated early into research design, rather than added as a dissemination step at the end of a project.

This also raises questions about how research impact is assessed. Traditional metrics such as citations may insufficiently capture the practical influence of transfer artefacts, calling for complementary evaluation approaches.

6 Conclusion and Outlook

This paper argued that closing the gaps between research and practice in ways of working requires rethinking transfer as a core research concern. Using the GMA Canvas as an illustrative case, it demonstrated how research-based insights can be translated into a practitioner-oriented artefact without sacrificing theoretical grounding.

Future work may explore how such transfer-oriented design principles can be systematically integrated into research projects on ways of working and how alternative dissemination formats may complement traditional academic publications. In particular, future work could investigate comparative cases, measure transfer outcomes more systematically, and explore how transfer artefacts evolve over time in different organizational contexts.

References

1. Pohl, K., Broy, M., Daembkes, H., Hönniger, H.: Technology transfer concepts. In: *Advanced Model-Based Engineering of Embedded Systems*, pp. 241–250. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48003-9_13
2. Cascio, W.F., Montealegre, R.: How technology is changing work and organizations. *Annu. Rev. Organ. Psych. Organ. Behav.* **3**(1), 349–375 (2016)
3. Cheung, I.W.: Plain language to minimize cognitive load: a social justice perspective. *IEEE Trans. Prof. Commun.* **60**(4), 448–457 (2017)
4. Clarke, P., O’connor, R.V.: The situational factors that affect the software development process: towards a comprehensive reference framework. *Inf. Softw. Technol.* **54**(5), 433–447 (2012)
5. Comelli, G., Von Rosenstiel, L.: *Führung durch Motivation: Mitarbeiter für Unternehmensziele gewinnen*. Vahlen (2011)
6. Cornish, F., et al.: Participatory action research. *Nat. Rev. Methods Primers* **3**(1), 34 (2023)
7. Diebold, P.: *Agile Practice Experience Repository for Process Improvement*, Fraunhofer Verlag. Ph.D. thesis (2020)
8. Diebold, P.: *Transformation zur individuellen Agilität-die agile Potenzialanalyse*. Springer, Cham (2022). <https://doi.org/10.1007/978-3-658-39412-7>
9. Heuer, A., Diebold, P., Bandyszak, T.: Supporting technology transfer by providing recommendations for writing structured guidelines. In: *Software Engineering (Workshops)*, pp. 47–56 (2014)
10. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. *MIS Q.* **28**(1), 75–105 (2004)
11. Nikolaidis, E., Mourelatos, Z.P., Pandey, V.: *Design Decisions Under Uncertainty with Limited Information*. CRC Press, Boca Raton (2011)
12. Schwartz, B., Ben-Haim, Y., Dacso, C.: What makes a good decision? robust satisficing as a normative standard of rational decision making. *J. Theory Soc. Behav.* **41**(2), 209–227 (2011)
13. Straub, D., Ang, S.: Editor’s comments: rigor and relevance in is research: Redefining the debate and a call for future research. *MIS Quart.* iii–xi (2011)

14. Tsui, M.s., O'Donoghue, K., Boddy, J., Pak, C.M.: From supervision to organisational learning: a typology to integrate supervision, mentorship, consultation and coaching. *Br. J. Soc. Work* **47**(8), 2406–2420 (2017)
15. Vetro, P.D., Fernandez, A.: An exploratory study on technology transfer in software engineering. In: *Proceedings of the 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) Proceedings*, pp. 86–95. (2015)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Closing Research–Practice Gaps through Action Research: Empowering Teams to Strengthen their Team Effectiveness

Anna Rook¹ , Bente Schreiter¹ , and Christiaan Verwijs² 

¹ West Coast University of Applied Sciences, Fritz-Tiedemann-Ring 20, 25746 Heide, Germany
{annroo30,bensch41}@student.fh-westkueste.de

² The Lucid Institute for Organizational Science (LIOS), Parelmervlinder 8, 3544DH Utrecht,
The Netherlands
christiaan.verwijs@lucidinstitute.org

Abstract. Bridging the gaps between research and practice remains a persistent challenge in software engineering and agile teamwork. Empirical insights often reach practitioners too late, fail to translate into team-specific contexts, or lack actionable guidance. This research-in-progress paper addresses these challenges by systematically closing two research–practice gaps: the time gap through real-time interaction between data and practice and the transfer gap through team-specific application of insights. The paper contributes on two complementary levels. First, it proposes a concrete pathway for closing these gaps by enabling teams to autonomously develop their ways of working through real-time, theory-informed feedback. Second, it demonstrates how action research can be systematically operationalized to both study and support this process. Building on prior research, the agile mindset is conceptualized as a multidimensional construct at the team level and examined using a structural equation model (SEM) in relation to team effectiveness. To date, 141 team-level datasets have been collected. Teams receive immediate automated feedback supporting pattern recognition, reflective dialogue, and continuous improvement. External stakeholders and internal promoters may be included to enable 360° evaluation. Repeated measurements allow teams to reassess progress and complete successive action research cycles. Preliminary findings indicate that (a) the agile mindset can be represented as a four-dimensional construct at the team level, and (b) it significantly and positively predicts team effectiveness. These results provide empirical support for the agile mindset as a team-level construct and highlight its relevance for improving team effectiveness in agile environments.

Keywords: Agile Mindset · Team Effectiveness · Action Research

1 Introduction

Despite a growing number of empirical studies on agile teamwork (e. g [1–4].) a persistent gap between research and practice remains, which considerably limits the practical value of scientific insights [5, 6]. In this context, two enduring research–practice gaps can be

conceptualized: the *time gap* and the *transfer gap* [6]. From a practical perspective, these gaps have tangible consequences: teams are required to make rapid decisions under uncertainty and often rely on intuition or locally available heuristics, while research-based evidence is insufficiently utilized. Action research offers a promising approach to bridging these gaps by closely linking knowledge generation and intervention with practice [7]. In this way, reciprocal synergies emerge, as researchers gain access to real organizational cases, valuable empirical data, and practice-oriented feedback, enabling their research to gain depth beyond theoretical models [5]. At the same time, teams and their stakeholders are understood as active co-creators who continuously develop their work through iterative cycles of hypothesis formation, action, observation, and reflection. This approach enables the generation of timely, context-specific, and action-relevant insights while simultaneously contributing to theory development. Nevertheless, further concrete examples are needed to illustrate how action research can effectively close these gaps. Moreover, recent literature increasingly addresses the importance of the agile mindset for performance and well-being in complex work environments (e. g. [2, 8, 9].) whereas quantitative research is still at an early stage. In addition, research on the agile mindset at the team level remains highly limited. Against this background, the following research question (*RQ*) is examined: *How does the agile mindset of teams influence their effectiveness?* This research-in-progress paper makes two novel contributions to the existing literature: The construct of the agile mindset is examined quantitatively at the team level, thereby extending the predominantly individual-level perspective. The nomological network of the agile mindset is systematically expanded by examining team effectiveness as an additional outcome variable [10].

For this purpose, the paper is structured as follows. A theoretically grounded structural equation model (SEM) is developed that conceptualizes the agile mindset as a multidimensional construct at the team level and examines its relationship with team effectiveness. In the methodological section, the quantitative SEM is tested through a practical platform-enabled action research approach, thereby systematically addressing existing research–practice gaps. In the results section, it is illustrated how this approach contributes to closing the research–practice gap through real-time interaction between data and practice and through the team-specific application of insights. In addition, the empirical results regarding the multidimensional structure of the agile mindset and its relationship with team effectiveness are presented. Finally, directions for future data collection and analysis are outlined.

2 Theoretical Background and Hypotheses

In the following, hypotheses are derived from prior research, which was iteratively developed through an action research approach.

In software development, agility is associated with sustainable performance and team effectiveness. Prior studies indicate that an agile mindset relates to organizational performance [11], job satisfaction and long-term effectiveness [12]. An agile mindset supports a people-, customer-, value-, and solution-oriented way of working by enabling flexibility and adaptation to changing requirements [13]. It reflects attitudes toward learning, collaboration, customer co-creation, and empowered self-guidance [11] which

evolve over time [14]. Building on prior research, the agile mindset can be conceptualized as a construct comprising four distinct but interrelated dimensions [11]: *Learning spirit* refers to a positive attitude toward seeking new experiences and acquiring knowledge in order to adapt to changing conditions. It reflects openness, curiosity, and a proactive willingness to explore unfamiliar situations and continuously learn.

Collaborative exchange captures a team member’s orientation toward knowledge sharing, feedback, and transparent collaboration in solving complex problems. Transparency, as a key principle of agility [15], enables individuals to better assess situations and actively contribute to collective decision-making. Individuals high in this dimension integrate diverse perspectives, share knowledge, and actively seek input from others. *Customer co-creation* describes a strong orientation toward customer value by actively involving stakeholders early in development and work processes. Through this continuous integration, teams are able to respond more effectively to changes and align their decisions with optimal customer outcomes [16]. *Empowered self-guidance* refers to an individual’s tendency toward self-organization, autonomous decision-making, and ownership of work processes [17]. It includes reflecting on one’s own work, identifying deviations from goals, and proactively developing and implementing new strategies based on insights gained. While these dimensions have been identified in prior research at the individual level, it remains unclear whether they can be empirically distinguished at the team level. Therefore, it is necessary to examine whether the proposed four-dimensional structure of the agile mindset can be empirically identified at the team level. Accordingly, the following hypothesis (H) is proposed: *H1: The agile mindset consists of four dimensions (a-d) at the team level.* Team effectiveness refers to the extent to which a team’s outcomes meet expected quality standards [18] and includes both external and internal dimensions, e. g. stakeholder satisfaction, reflecting evaluations by clients, customers, or users, and team morale, representing the team’s internal appraisal of outcomes. In agile teams, responsiveness, stakeholder concern, continuous improvement, team autonomy, and management support predict morale and stakeholder satisfaction [4]. However, the contribution of the agile mindset to team effectiveness remains insufficiently understood, yet it may provide highly relevant insights for increasing the effectiveness of agile software development teams. This leads to the following hypothesis: *H2: The team agile mindset is positively related to team effectiveness.* Based on these deviations, the research model is built (Fig. 1).

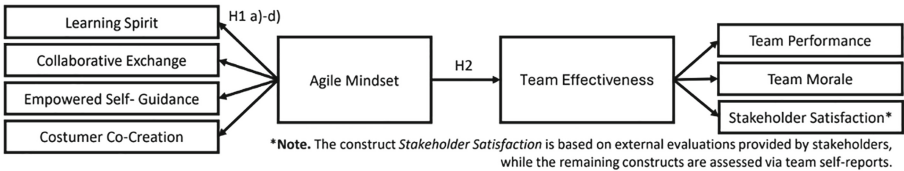


Fig. 1. Research model based on Eilers and Verwijns [10].

3 Methodological Background

3.1 Measurements

To investigate the team agile mindset model, a SEM approach was employed. All constructs were operationalized as reflective measures and assessed using a 7-point Likert scale (1 = *Completely disagree* to 7 = *Completely agree*). Team agile mindset was measured across four dimensions based on scales from Eilers et al. [11] and adapted for team-level analysis [9]: *customer co-creation* (3 items; e.g., “Our team strives to understand what customers need by interacting with them directly”), *collaborative exchange* (3 items; e.g., “Our team members like to share their thoughts when someone in the team seeks advice”), *learning spirit* (3 items; e.g., “Members of this team like to explore new ways to better complete our work”), and *empowered self-guidance* (2 items; e.g., “It is important for our team to contribute to setting goals for the work we do”). The construct *Team Effectiveness* was measured using multiple outcome dimensions capturing both internal team functioning and stakeholder-related results [3]: *team performance*, *team morale* (both team-rated), and *stakeholder satisfaction* (stakeholder-rated). *Team morale* (2 items; e.g., “I am proud of the work that I do for this team”; adapted from Schaufeli et al. [19]) captures the team’s internal evaluation of its functioning. *Team performance* (2 items; e.g., “This team achieves its assigned performance goals”; adapted from Aube and Rousseau [20]) reflects the extent to which teams meet their objectives. *Stakeholder satisfaction* (3 items; e.g., “Stakeholders are generally happy with what this team delivers”; based on Verwijns and Russo [4]) captures external evaluations of team outcomes. All scales demonstrated satisfactory internal consistency (Cronbach’s α).

3.2 Data Collection with Action Research

Data collection is ongoing; to date, $N = 141$ responses across 54 teams have been obtained. The participants represented a wide range of industries, with the largest proportions coming from the technology sector (29.1%) and the public sector (16.3%). The participating teams ranged in size from 3 to 16 members and were primarily located in Western Europe (62.4%), followed by East Asia (14.9%) and North America (8.5%). The most common team types were those focused on maintaining existing products (31.9%) and new product development (24.1%). The study follows an action research design based on repeated cycles of diagnosis, reflection/action, and evaluation [7], aiming to generate context-sensitive empirical insights while simultaneously supporting targeted change in ongoing team practice. The approach is explicitly designed as a multi-level framework, allowing interventions and data collection at the team, team-of-teams, and organizational levels. By applying a consistent cyclical logic across these levels, developmental processes are embedded within the broader organizational context rather than confined to individual teams. The researchers act as independent academic partners responsible for systematic observation, analysis, theory-informed recommendations, and structured documentation, while collaborating closely with the team to establish short feedback loops, identify ambiguities, and uncover missing elements. The participants consist of diverse practitioner teams who contribute challenges, provide the necessary context, and retain full decision-making authority over the implementation and adoption

of the proposed measures. The digital platform Columunity [21] provides the technical and procedural infrastructure for implementing this cycle:

Diagnosis Phase. Initially, teams complete standardized surveys on team processes and outcomes and receive immediate analyses that identify patterns, strengths, and areas for development. These analyses provide the empirical basis for subsequent steps and are complemented by references to relevant academic studies and theoretical models.

Reflection and Intervention Phase. In this phase, teams interpret results, formulate improvement hypotheses, and implement actions supported by theory-based recommendations and facilitation guidelines, provided by the platform.

Evaluation Phase. After a defined period, data is collected again and compared with the previous measurements to assess the effects of the implemented actions. The insights gained then feed into the next cycle of diagnosis and intervention.

This iterative process enables the continuous integration of theory-driven research and practical development. All relevant stakeholder groups with an interest in solving a problem or improving a process may be involved. Through this integration of theory-driven research and practical intervention, the proposed action research design directly engages with the key challenges at the research–practice interface.

4 Preliminary Results

4.1 Closing the Research-Practice Gaps with Action Research

This section explains how the proposed action research design addresses the two research–practice gaps [5, 6].

Time Gap. While methodologically sound research often demands extended periods for data collection, validation, and publication, organizational practice usually cannot afford to wait for fully matured results. This action research approach via the platform Columunity addresses this challenge by enabling the continuous, theory-guided examination of research models within ongoing practice. At early stages of the research models, relationships between constructs are not yet empirically confirmed. However, as data collection progresses across multiple teams and over time, the empirical basis of the models steadily improves, thereby increasing their validity. The models are not treated as static artifacts; instead, they are successively adapted on the platform based on the accumulating data. This iterative refinement allows emerging empirical insights to be incorporated without waiting for the completion of long-term, comprehensive studies. In parallel, intermediate findings and reflections are regularly shared and discussed within the broader community through various formats such as blog posts, updates, meet-ups, and podcasts. This continuous dissemination supports timely sensemaking and collective learning while maintaining transparency about the evolving state of the models. Whereas fully comprehensive research processes would be too slow to meet the needs of practice, the cyclical interaction between theory and practice enabled by this action research approach facilitates rapid feedback, ongoing dialogue, and joint knowledge

development. In doing so, it contributes to narrowing the time gap by allowing scientifically informed insights to become available and actionable within practice much earlier, while still preserving a trajectory toward increasing rigor over time.

Transfer Gap. This gap often arises because research outputs are communicated in formats that are difficult to translate into concrete actions or because practitioners are insufficiently involved in the knowledge creation process. This action research approach addresses the transfer gap by directly integrating practitioners—specifically teams—into the research process itself. Rather than positioning teams as passive recipients of research results, this approach involves them as active contributors in the formulation of research questions, the application of theory-based models, and the interpretation of findings. As teams work with theoretical constructs in their everyday contexts, abstract concepts are translated into concrete experiences and actionable insights. This close coupling of knowledge generation and application increases the likelihood that research outcomes are not only understood but also meaningfully adopted in practice. In addition, research results are not solely disseminated through traditional academic channels but are translated into practice-oriented artifacts such as reflection prompts, guidelines, and team-level interventions. These artifacts are iteratively refined based on feedback from ongoing use, thereby functioning as boundary objects that facilitate knowledge transfer between academic and practitioner communities. Through this participatory, iterative, and artifact-oriented approach, the action research design reduces barriers to knowledge transfer and fosters bidirectional learning. Consequently, the transfer gap is narrowed by ensuring that theoretical insights are continuously shaped by practice and rendered usable within real organizational settings.

4.2 Preliminary Research Results Regarding the Agile Mindset Model

To address the RQ, the preliminary analyses reveal the following results. To H1, a confirmatory factor analysis (CFA) was conducted to examine the four-dimensional structure of the team agile mindset. The initial model did not show an adequate fit to the data. Therefore, the measurement model was iteratively refined by removing items with cross-loadings and problematic error variances, resulting in a final set of 11 items [22]. The final model demonstrated a good fit to the data, ($\chi^2(71) = 115.26$, $p < .001$, CFI = .969, TLI = .970, RMSEA = .067, SRMR = .045). All constructs showed acceptable to high internal consistency, with Cronbach's alpha values ranging from .87 to .93. Taken together, these results provide empirical support for the proposed four-dimensional structure of the agile mindset at the team level. To test H2, a structural equation model was estimated to examine the relationship between team agile mindset and team effectiveness. The structural model showed a good fit to the data ($\chi^2(144) = 226.35$, $p < .001$, CFI = .959, TLI = .946, RMSEA = .064, SRMR = .061). The results indicate that the agile mindset of teams has a strong and positive effect on team effectiveness ($\beta = .781$, $p < .001$). Thus, H2 is supported.

5 Next Research Steps

Future research should extend the nomological network of the agile mindset by examining relevant determinants, particularly work design and leadership. Accordingly, the following hypotheses are proposed. *Work design* refers to the structuring of tasks and enabling conditions such as autonomy and feedback and has been linked to motivation, learning, and work engagement [23, 24]. In agile contexts, work design can enable or constrain the development of an agile mindset [11]. Empirical evidence indicates that team autonomy fosters self-organized and adaptive work [25] and that a strong value focus makes a central contribution to the effectiveness of agile teams [4]. *H3: The work design is positively related to the team agile mindset.* *Leadership* represents a key contextual factor shaping team processes over time [26]. In agile settings, leadership shifts toward enabling self-organization [27], with leaders acting as role models and creating conditions that support agile thinking and behavior [17]. Moreover, continuous management support has been associated with agile team effectiveness [4], while insufficient support may hinder agility [28]. *H4: Leadership is positively related to team agile mindset.*

References

1. Brede Moe, N., Dingsoyr, T., Dyba, T.: Understanding Self-Organizing Teams in Agile Software Development, pp. 76–85 (2008). <https://doi.org/10.1109/ASWEC.2008.4483195>
2. Miler, J., Gaida, P.: On the agile Mindset of an effective team – An Industrial Opinion Survey, pp. 841–849 (2019). <https://doi.org/10.15439/2019F198>
3. Verwijns, C., Eilers, K., Russo, D.: Do agile scaling approaches make a difference? An empirical comparison of team effectiveness across popular scaling approaches. *Empir Software Eng.* **29** (2024). <https://doi.org/10.1007/s10664-024-10481-5>
4. Verwijns, C.E., Karen, R.D.: A theory of scrum team effectiveness. *ACM Trans. Softw. Eng. Methodol.* **32**, 1–51 (2023). <https://doi.org/10.1145/3571849>
5. Barroca, L., Sharp, H., Salah, D., Taylor, K., Gregory, P.: Bridging the gap between research and agile practice: an evolutionary model. *Int. J. Syst. Assur. Eng. Manag.* **9**, 323–334 (2018). <https://doi.org/10.1007/s13198-015-0355-5>
6. Neumann, M., Schön, E.-M., Senapathi, M., Rauschenberger, M., Da Silva Silva, T.: Towards a closer collaboration between practice and research in agile software development workshop. In: *A Summary and Research Agenda*, vol. 561, pp. 91–97. https://doi.org/10.1007/978-3-032-05799-0_9
7. Avison, D.E., Lau, F., Myers, M.D., Nielsen, P.A.: Action research. *Commun. ACM.* **42**, 94–97 (1999). <https://doi.org/10.1145/291469.291479>
8. Geffers, K., Bretschneider, U., Karen, E.: Exploring individuals’ psychological factors as predictors of workforce agility in software development teams. In: *ICIS 2024 Proceedings* (2024)
9. Augner, T., Schermuly, C.C.: Beyond a buzzword: the agile mindset as a new research construct in organizational psychology. *JMP.* **40**, 274–288 (2025). <https://doi.org/10.1108/JMP-04-2024-0261>
10. Verwijns, C., Eilers, K.: How a Team-Level Agile Mindset Shapes Team Effectiveness. (2025), <https://medium.com/the-liberators/how-a-team-level-agile-mindset-shapes-team-effectiveness-880f9b2b8f89>
11. Eilers, K., Peters, C., Leimeister, J.M.: Why the agile mindset matters. *Technol. Forecast. Soc. Chang.* **179**, 121650 (2022). <https://doi.org/10.1016/j.techfore.2022.121650>

12. Peters, C., Simmert, B., Eilers, K., Leimeister, J.M.: Future Organization Report 2021. Institute of Information Management at University of St.Gallen and Campana & Schott Business Services GmbH, St.Gallen, Switzerland and Frankfurt (2021)
13. Özkan, N., Eilers, K., Gök, M.Ş.: A Literature Review Based Insight into Agile Mindset through a Lens of Six C's Grounded Theory Model, vol. 504, pp. 261–282. https://doi.org/10.1007/978-3-031-61657-0_13
14. Schwarz, N.: Attitude construction: evaluation in context. *Soc. Cogn.* **25**, 638–656 (2007). <https://doi.org/10.1521/soco.2007.25.5.638>
15. Schwaber, K., Sutherland, J.: The scrum guide. The rules of the game. (2020) <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>
16. Lorenz, R., Bäckert, L., Heck, J.: Principles of operational agility: a case study of a Swiss telecommunication company. *DU.* **74**, 173–186 (2020). <https://doi.org/10.5771/0042-059X-2020-2-173>
17. Schreiter, B., Rook, A., Eilers, K., Warszta, T.: Wie Führungskräfte ein agiles Mindset fördern können. *PERSONALQuarterly*, 40–47 (2025)
18. J. R. Hackman: The Design of Self-Managing Work Groups (1976)
19. Schaufeli, W.B., Bakker, A.B., Salanova, M.: The measurement of work engagement with a short questionnaire. *Educ. Psychol. Meas.* **66**, 701–716 (2006). <https://doi.org/10.1177/0013164405282471>
20. Aubé, C., Rousseau, V.: Team goal commitment and team effectiveness: the role of task interdependence and supportive Behaviors. *Group Dyn. Theory Res. Pract.* **9**, 189–204 (2005). <https://doi.org/10.1037/1089-2699.9.3.189>
21. Columinity: Improve your teams. Boost your business (2025), <https://columinity.com>
22. Hair, J.F., Risher, J.J., Sarstedt, M., Ringle, C.M.: When to use and how to report the results of PLS-SEM. *EBR.* **31**, 2–24 (2019). <https://doi.org/10.1108/EBR-11-2018-0203>
23. Parker, S.K.: Beyond motivation: job and work design for development, health, ambidexterity, and more. *Annu. Rev. Psychol.* **65**, 661–691 (2014). <https://doi.org/10.1146/annurev-psych-010213-115208>
24. Hackman, J., Oldham, G.R.: Motivation through the design of work: test of a theory. *Organ. Behav. Hum. Perform.* **16**, 250–279 (1976). [https://doi.org/10.1016/0030-5073\(76\)90016-7](https://doi.org/10.1016/0030-5073(76)90016-7)
25. Junker, T.L., Bakker, A.B., Gorgievski, M.J., Derks, D.: Agile work practices and employee proactivity: a multilevel study. *Hum. Relat.* **75**, 2189–2217 (2022). <https://doi.org/10.1177/00187267211030101>
26. Yukl, G.: Effective leadership behavior: what we know and what questions need more attention. *AMP.* **26**, 66–85 (2012). <https://doi.org/10.5465/amp.2012.0088>
27. Manz, C.C., Sims, H.P.: Leading workers to Lead themselves: the external leadership of self-managing work teams. *Adm. Sci. Q.* **32**, 106 (1987). <https://doi.org/10.2307/2392745>
28. Goncalves, D., Bergquist, M., Bunk, R., Alänge, S.: The Influence of Cultural Values on or-Ganizational Agility. In: Americas Conference on Information Systems, p 1–10 (2019)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Human-AI Collaboration in Agile Teams



AI-Augmented Agile Teams: A Research Agenda and Summary of the Human-AI Collaboration Workshop

Nils Brede Moe¹(✉)  and Viktoria Stray^{1,2} 

¹ SINTEF, Trondheim, Norway
nilsm@sintef.no, stray@ifi.uio.no
² University of Oslo, Oslo, Norway

Abstract. Agile teams are rapidly adopting AI agents in everyday work, reporting increased productivity and faster feedback cycles. Simultaneously, the integration of AI raises new socio-technical issues concerning coordination, trust, roles, quality, and governance. To better understand these challenges, we organized the first International Human-AI Collaboration in Agile Teams workshop at XP 2026, with more than 40 participants from academia and industry. This paper reports on the outcomes of the workshop, synthesizing insights from interactive group discussions and collaborative analysis. The findings highlight how AI reshapes agile teamwork and workflows, introducing new human-AI collaboration patterns, shifting skills and roles, and raising challenges related to leadership, governance, and value assessment. A key emerging concept is cognitive debt, referring to the gradual loss or weakening of shared understanding in teams relying on AI-generated outputs. Based on these findings, we present a research agenda structured around eight key themes, including teamwork, workflows, autonomy, skills, value measurement, and adoption practices. The results emphasize that integrating AI into agile software development teams is not only a technical challenge but a socio-technical transformation requiring new practices, competencies, and organizational support.

Keywords: Teamwork · Agile · Agentic coding · Autonomy · copilot · Claude

1 Introduction

Generative AI (GenAI) systems such as ChatGPT, Claude, and GitHub Copilot are rapidly being adopted in software development, driven by expectations of substantial productivity gains [11, 16]. In agile settings, teams are increasingly incorporating not only AI assistants but also more autonomous agents into everyday activities, including planning, design, coding, testing, documentation, and analytics. The growing use of AI assistants and autonomous agents across the agile workflow promises faster feedback cycles and improved efficiency, while

also introducing new forms of human–AI collaboration [13] and reshaping how work is performed within agile teams.

Building on this trend, recent advances in agentic coding move beyond isolated code generation to more integrated, team-like collaboration across agile development activities. These approaches model AI systems as specialized agents that assume roles similar to those in agile teams (e.g., product management, development, and testing), working iteratively and facilitating continuous feedback and alignment with user needs [10]. The shift from code generation to multi-agent collaboration positions AI less as a tool and more as a kind of AI teammate working within structured workflows. While these approaches demonstrate the potential of AI-augmented teamwork, it remains to be seen how these agentic systems will be integrated with real-world agile practices and how they will affect coordination, roles, and shared understanding within teams [5].

Early empirical evidence from industry suggests that the productivity gains from AI adoption are uneven and not straightforward to realize in practice [7]. Recent longitudinal evidence indicates that perceived productivity benefits from GitHub Copilot may not be reflected in conventional activity metrics such as commits or lines changed [15]. Many organizations have yet to observe measurable business value from their AI investments [12], indicating that successful adoption requires more than access to tools. Instead, it demands changes in how work is organized, coordinated, and learned within teams. This challenge is further complicated by the evolving nature of GenAI technologies, which cannot be treated as stable artifacts but rather as objects of ongoing experimentation and negotiation in practice [18]. In practice, organizations often struggle to translate technological capabilities into improved development flow or sustained productivity gains [2].

Although recent empirical work has begun to examine how generative AI affects developers’ everyday workflows, including solo and pair programming with ChatGPT and GitHub Copilot [17], there is still limited understanding of how such tools reshape agile teamwork at the collective level, including coordination, shared practices, roles, and value creation. This challenge is amplified by the dynamic nature of software development, where AI agents and collaboration patterns may need to continuously adapt to evolving requirements and project conditions [5].

To address this gap, we organized the Human–AI Collaboration in Agile Teams workshop at XP 2026. The workshop brought together more than 40 experts from academia and industry and combined short presentations with interactive group discussions. The goal was to explore how AI affects everyday agile activities and to collaboratively identify emerging challenges, practices, and research directions.

This paper reports on the outcomes of the workshop. Based on the discussions, we identify key themes characterizing human–AI collaboration in agile teams, including changes in workflows and team dynamics, evolving roles and competencies, challenges related to governance and value assessment, and the emergence of cognitive debt as a potential risk. Furthermore, we synthesize a

research agenda that outlines key questions for future empirical and conceptual work. By providing an empirically grounded overview of current challenges and opportunities, this paper contributes to the growing body of knowledge on AI-augmented software teams and offers direction for both researchers and practitioners in the XP community.

2 Research Papers and Industry Reports

The workshop included five presentations. Three were research contributions that had been peer-reviewed and accepted by the program committee, while two were industry experience reports invited to ground the discussion in current practice. Together, the presentations provided complementary perspectives on how AI is entering agile software development, from individual use of generative AI tools to agentic systems, AI-supported testing, and organizational models for AI-augmented development.

The first research contribution, by El Outmani et al. [3], examined human-AI collaboration in agile regression testing through an industrial case study at Hacon, a Siemens company. The second research contribution, by Gundelsby et al. [4], investigated how individual use of generative AI affects collaboration in agile teams. Based on thirteen interviews with software and technology professionals, the study found that GenAI use was primarily individual rather than shared at the team level. Participants often asked AI before asking colleagues, which could reduce interruptions and support individual problem solving, but also risked weakening mentoring, collective learning, and shared understanding. The third research contribution, by Hinderks et al. [6], reframed AI integration as an organizational design problem rather than a tool adoption problem. The paper proposed a three-layer model for AI-Dev-Systems, defined as configured environments of AI agents, tools, workflows, and knowledge bases that support systematic software development. In this model, System Teams design AI-Dev-Systems, Platform Teams maintain and evolve them, and Product Teams develop software within these environments.

The two industry experience reports complemented the research contributions by grounding the discussion in current practice. One report described AI-assisted model-based testing in the company Testify AS and showed how AI can support earlier feedback and improved collaboration, while also introducing questions about ownership, supervision, and changing competence needs. The other addressed the “AI paradox” based on experiences from Team Rockstars, where individual productivity improvements do not necessarily translate into team-level benefits. Instead, increased coordination overhead, fragmented AI use, and reduced shared understanding may weaken collective effectiveness even when individuals work faster.

3 Workshop Findings

Following the presentations, the participants divided into four facilitated groups to engage in structured discussions. Participants could choose among the fol-

lowing topics, or suggest their own: 1) How AI Changes the Workflow, 2) From Potential to Value: Why AI Does (Not) Deliver, 3) Skills, Roles, and Careers in AI-Augmented Teams, and 4) Leadership and Work in the Age of AI. Each group discussion was guided by a predefined set of questions related to the topic.

The discussions followed a multi-step facilitation process. First, participants individually reflected on the topic and wrote down ideas, observations, and concerns on sticky notes. Next, participants presented their reflections within the group, followed by collaborative discussion and clustering of related themes and perspectives. Each group presented its main findings to the workshop participants in a plenary session. Based on the discussions, participants were also encouraged to formulate research questions and identify areas for future work.

The workshop facilitators collected all sticky notes and discussion material generated during the thematic discussions and research agenda session. The material was subsequently reviewed and consolidated by the authors through iterative thematic analysis, resulting in recurring socio-technical themes and associated research questions characterizing human–AI collaboration in agile teams.

3.1 AI Reshapes Teamwork and Workflows

A recurring theme across the groups was that AI is changing everyday development practices, but not in a uniform way. Participants described shifts across the entire agile workflow, including planning, development, testing and reviewing (see Fig. 1). AI was often associated with faster completion of individual tasks. However, participants were more cautious about whether this made the overall agile development process more efficient.

Rather, the discussions suggest that AI changes where work takes place and what kind of work becomes necessary. In some cases, effort appears to move from producing an artifact to checking, adapting, integrating, or explaining it. This creates new coordination needs within teams, as well as new dependencies between tools, roles, and established ways of working. Several participants pointed out that AI-generated outputs may save time when solving tasks, while also introducing validation overhead and integration challenges elsewhere.

3.2 Emerging Human–AI Teaming Dynamics

Participants did not describe AI only as a tool that supports existing work practices. In several discussions, AI was instead talked about as something that actively shapes how developers approach tasks, make decisions, and interact with each other. This raised questions about trust, transparency, and the conditions under which human judgement should remain central.

One concern was that AI was often used individually by team members. This supports individual productivity, but it can also make parts of the work less visible to the rest of the team. Assumptions, decisions, or generated artifacts may not be shared in the same way as before. The material therefore points to a tension between individual efficiency and collective ownership. AI can help

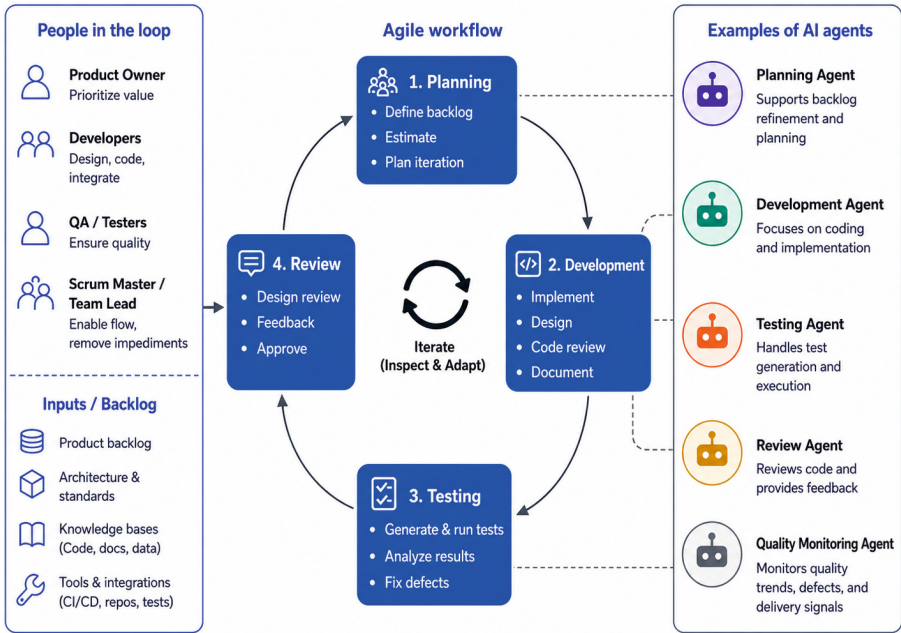


Fig. 1. Human–AI collaboration across the agile workflow.

developers move faster, but it may also make it harder for teams to maintain a shared understanding of the system and of the work being done.

3.3 Shifting Skills, Roles, and Career Paths

AI is reshaping what counts as important competence in software teams. Participants emphasized the need for system-level understanding, a focus on business value, orchestration skills, and the ability to evaluate AI-generated outputs critically. The shift was therefore not described simply as a reduction in technical skill needs.

Further, participants questioned how newcomers will develop foundational expertise if AI increasingly supports the tasks through which such expertise has traditionally been learned. The concern is not only whether junior developers can use AI effectively, but whether they will develop the judgement needed to understand and take responsibility for AI-generated outputs. In this sense, AI adoption may affect both role expectations and longer-term career paths in software development.

3.4 Leadership, Governance, and Organizational Alignment

Leadership and organizational support were described as important conditions for responsible AI adoption. Participants emphasized that AI use involves more

than tool choice. It also raises legal, ethical, financial, and managerial questions. Teams need clearer boundaries for acceptable use.

At the same time, participants stressed that responsibility must remain with humans. This creates a need for governance that supports experimentation without leaving AI adoption entirely to individual initiative. This cuts across all stages of the workflow in Fig. 1. Regardless of which agents support which stage, accountability stays with the humans in the loop.

3.5 Value, Quality, and Measurement Challenges

Although AI was frequently linked to productivity, participants questioned how its value should be assessed. Faster task completion was not necessarily seen as the same as better outcomes. Participants raised concerns about token costs, customer value, artifact quality, maintainability, and longer-term consequences such as technical debt [1], process debt [8], and cognitive debt [14].

Cognitive debt is a team-level, project-level property reflecting the erosion of shared understanding across a software system over time [14]. In the workshop discussions, this concern appeared when participants described how reliance on AI-generated outputs that are not fully understood, validated, or discussed collectively may gradually weaken shared understanding within teams. Participants noted that while individual productivity gains may be real, they are often difficult to measure and may come at the cost of reduced transparency, learning, and long-term quality. These discussions highlight the need for evaluation approaches that go beyond individual productivity metrics.

3.6 Adoption, Learning, and Practice Evolution

Teams are experimenting with how AI could be integrated into their work and workflow, and practices seemed to develop through trial and error. Effective use depends on continuous learning, feedback, and the sharing of experience within and across teams.

At the same time, participants noted several barriers to such learning. These included lack of time, uncertainty about appropriate use, and resistance to change. AI adoption therefore appears less like a one-time implementation and more like a gradual evolution of work practices. Teams need shared routines for experimentation and a common language to discuss how AI is used, what kinds of outputs can be trusted, and how human oversight should be organized. Earlier work on attracting and retaining software engineers has shown that dedicated time for learning, such as a recurring “competence day”, can support competence development and team learning in software organizations [9]. Comparable mechanisms may be needed in AI-augmented teams, where the relevant competencies are still evolving.

Table 1. Research themes and questions identified from the workshop.

Theme	Research question	Focus
Teamwork & Collaboration	How does AI reshape teamwork, coordination, and communication in agile teams?	Collaboration patterns, human–AI interaction, and team dynamics.
Workflows & Practices	How does AI change software development workflows and agile practices?	Development lifecycle, task distribution, and process adaptation.
Autonomy & Governance	How should teams balance human autonomy, AI autonomy, and organizational control?	Decision-making, accountability, and governance models.
Skills & Roles	What skills, roles, and career paths are needed in AI-augmented teams?	Competence development, role evolution, and implications for education.
Value & Measurement	How can organizations measure the value, quality, and productivity impact of AI?	Metrics, evaluation frameworks, and long-term effects.
Adoption & Learning	How should teams adopt and learn to use AI effectively?	Onboarding practices, team learning, and experimentation.
Conditions for Success	When does AI enhance teamwork and when does it create risks?	Contextual factors, boundaries, and unintended consequences.
Leadership & Organization	What leadership and organizational practices enable effective AI integration and scaling of agentic coding?	Change management, culture, and organizational design.

3.7 Cross-Cutting Tensions

Taken together, the themes reveal several recurring tensions related to AI adoption in agile teams. Participants described tensions between efficiency and quality, autonomy and control, individual productivity and team alignment, and experimentation and standardization. These tensions appeared across multiple discussions and reflected differing assumptions about how AI should be integrated into agile work practices.

4 Research Agenda and Concluding Remarks

Building on the workshop findings, we synthesized a research agenda that captures key challenges and open questions related to human–AI collaboration in agile teams. The agenda is grounded in themes and questions identified through group discussions and reflects both research and industry perspectives. Table 1 presents the resulting themes and associated research questions as a starting point for future work.

Across the presentations and discussions, a common message emerged: AI in agile teams cannot be understood solely in terms of individual productivity or tool support. Instead, AI affects agile work at multiple levels, including how individuals solve problems, how teams coordinate and learn, how workflows are organized, and how development is governed at the organizational level. A recurring concern was the emergence of cognitive debt, where reliance on AI-generated outputs may gradually weaken shared understanding, learning, and collective ownership within teams.

The workshop also revealed recurring tensions between speed and quality, individual autonomy and team alignment, AI autonomy and human accountability, and experimentation and standardization. These tensions suggest that integrating AI into agile teams is not merely a technical challenge, but an ongoing socio-technical transformation requiring new practices, competencies, and governance mechanisms.

In conclusion, this paper reports on the first Human–AI Collaboration in Agile Teams workshop at XP 2026 and provides an empirically grounded overview of how AI is currently reshaping agile teamwork. Based on the discussions, we identified seven socio-technical themes and developed a research agenda to guide future empirical work and industry practice. Future research should develop shared approaches for assessing value, quality, and team-level productivity in AI-augmented settings, since current individual productivity metrics remain insufficient.

Acknowledgement. We are thankful to all workshop participants for engaging discussions and for generously sharing their time and insights. The research was supported by the Research Council of Norway through the Kairos project (grant 357147).

References

1. Besker, T., Martini, A., Bosch, J.: Software developer productivity loss due to technical debt—a replication and extension study examining developers’ development work. *J. Syst. Softw.* **156**, 41–61 (2019)
2. Bosch, J., Olsson, H.H.: Towards AI-driven organizations. In: Taibi, D., Smite, D. (eds.) SEAA 2025. LNCS, vol. 16083, pp. 280–295. Springer, Cham (2025). https://doi.org/10.1007/978-3-032-04207-1_19
3. El Outmani, M., et al.: Human–AI collaboration for scaling agile regression testing: an agentic-AI teammate from manual to automated testing. In: *Agile Processes in Software Engineering and Extreme Programming – Workshops: São Paulo, Brazil, April 8, 2026*. LNBIP, Springer, Cham (2026)
4. Gundelsby, J.H., Berntzen, M., Grøneng, C., Vigeland, D.: “I’ll Ask AI First”: how GenAI disrupts collaboration in agile teams. In: *Agile Processes in Software Engineering and Extreme Programming – Workshops: São Paulo, Brazil, April 8, 2026*. LNBIP, Springer, Cham (2026)
5. He, J., Treude, C., Lo, D.: LLM-based multi-agent systems for software engineering: literature review, vision, and the road ahead. *ACM Trans. Software Eng. Methodol.* **34**(5), 1–30 (2025)

6. Hinderks, A., Thomaschewski, J., Schön, E.M.: From AI assistants to AI-dev-systems: a three-layer model for human–AI collaboration in agile software development. In: *Agile Processes in Software Engineering and Extreme Programming – Workshops: São Paulo, Brazil, April 8, 2026*. LNBIP, Springer, Cham (2026)
7. Humlum, A., Vestergaard, E.: Large language models, small labor market effects. Technical report, National Bureau of Economic Research (2025)
8. Martini, A., Stray, V., Besker, T., Moe, N.B., Bosch, J.: Process debt: definition, risks, and management. *J. Software: Evol. Process* **37**(4), e70017 (2025). <https://doi.org/10.1002/smr.70017>
9. Moe, N.B., Stray, V., Smite, D., Mikalsen, M.: Attractive workplaces: what are engineers looking for? *IEEE Softw.* **40**(5), 85–93 (2023). <https://doi.org/10.1109/MS.2023.3276929>
10. Nguyen, M.H., Chau, T.P., Nguyen, P.X., Bui, N.D.: Agilecoder: dynamic collaborative agents for software development based on agile methodology. In: *2025 IEEE/ACM Second International Conference on AI Foundation Models and Software Engineering (Forge)*, pp. 156–167. IEEE (2025)
11. Nguyen-Duc, A., et al.: Generative artificial intelligence for software engineering—a research agenda. *Softw. Practice Exper.* **55**(11), 1806–1843 (2025). <https://doi.org/10.1002/spe.70005>
12. PwC: 29th global CEO survey: Leading through uncertainty in the age of AI (2026). <https://www.pwc.com/gx/en/issues/c-suite-insights/ceo-survey.html>
13. Rasheed, Z., Waseem, M., Kemell, K.K., Saari, M., Abrahamsson, P.: LLM-based multi-agent systems for code generation: a multi-vocal literature review. *arXiv preprint arXiv:2604.16321* (2026)
14. Storey, M.A.: From technical debt to cognitive and intent debt: Rethinking software health in the age of AI. *arXiv preprint arXiv:2603.22106* (2026)
15. Stray, V., Brandtzæg, E.G., Wivestad, V., Barbala, A., Moe, N.B.: Developer productivity with and without GitHub Copilot: a longitudinal mixed-methods case study. In: *Proceedings of the 59th Hawaii International Conference on System Sciences (HICSS 2026)* (2026). <https://doi.org/10.24251/HICSS.2026.880>
16. Stray, V., Hanssen, G.K., Barbala, A., Šmite, D., Stol, K.J.: What is generative AI good for? Introduction to the special issue on generative AI in software engineering. *Inf. Softw. Technol.* **187**, 107857 (2025). <https://doi.org/10.1016/j.infsof.2025.107857>
17. Stray, V., Moe, N.B., Ganeshan, N., Kobbenes, S.: Generative AI and developer workflows: how GitHub Copilot and ChatGPT influence solo and pair programming. In: *Proceedings of the 58th Hawaii International Conference on System Sciences (HICSS)* (2025). <https://doi.org/10.24251/HICSS.2025.883>
18. Ulfesnes, R., Barbala, A., Moe, N.B., Mikalsen, M.: The materialization of GenAI: examining the implementation of large language models in knowledge organizations. In: *Proceedings of the 59th Hawaii International Conference on System Sciences*. University of Hawaii at Mānoa, Honolulu, HI (2026). <https://doi.org/10.24251/HICSS.2026.765>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





“I’ll Ask AI First”: How GenAI Disrupts Collaboration in Agile Teams

Jan Henrik Gundelsby¹(✉) , Marthe Berntzen^{1,2} , Camilla Grøneng¹ ,
and Dan Vigeland¹

¹ Knowit, Universitetsgata 1, 0164 Oslo, Norway

{jhg,marthe.berntzen}@knowit.no, marthenb@ifi.uio.no

² University of Oslo, Gaustadalleen 23B, 0373 Oslo, Norway

Abstract. Agile teams depend on collaboration, collective learning, and shared accountability to function effectively. As team members increasingly use generative AI (GenAI) tools, these foundational dynamics may be disrupted. In this study, we investigate how individual human–AI collaboration affects agile teamwork through thirteen interviews with software and technology professionals. Our findings show that GenAI reshapes team collaboration and learning while enabling individuals to work across traditional role boundaries, but also reinforces the continued necessity of human oversight and team accountability. We argue that these dynamics challenge the notion of agile teams, and call for updated models of agile teamwork.

Keywords: Agile Teams · Human-AI Collaboration · Teamwork

1 Introduction and Background

GenAI changes teamwork in software engineering. Since the early 2000s, much commercial software has been developed by agile teams: small, cross-functional, self-organizing groups that collaborate closely with stakeholders to deliver working software in short, iterative cycles. The agile team has clear product ownership and autonomy to decide how work gets done. Team members engage in continuous communication, while consistently inspecting and adapting to improve both the product and their own process [9]. Importantly, the agile team is a *team*. As opposed to a group, which consists of externally managed individuals, a team works collectively toward a common goal by leveraging individual efforts and coordinating amongst themselves [5].

While some development teams are already using GenAI tools collaboratively to augment team capabilities [7], much human-AI collaboration is still conducted individually, potentially enabling single developers to carry out larger parts of the software development process independently [2]. As individual members of agile teams increasingly use GenAI to support their work, questions arise as to whether this human-AI collaboration leads to a re-individualization of work [1], and how this may influence agile teamwork.

During the AI and Agile workshop at the XP conference in 2025, challenges and research opportunities related to agile teams permeated the emerging research agenda. For instance, fragmented tool use, the potential threats by AI to team mental models due to reduced interaction, changes in learning patterns and process misalignment at the team level were among the prominent challenges [6].

The positive and negative implications of GenAI for team processes and learning are already being explored, such as increased employee efficiency, motivation and learning [15], but also potentially lower levels of collaboration in teams, with negative impact on employee affective affordances and well-being [3]. There are also organization-level challenges including underestimated overhead and limitations related to lack of tool integration with the development context [2]. All the while, companies are starting to introduce organization-level policies and strategies, moving AI use from the individual to the team and organizational level [2, 10]. However, much of the empirical research on GenAI to date has considered individual use, leaving a gap related to the impact of GenAI on workflows and team dynamics [6, 10].

A recent systematic mapping study shows that research on how GenAI affects agile team dynamics is still limited [8], leaving a gap related to the impact of GenAI on team workflows and dynamics [6, 10]. Among the topics studied are questions on how GenAI influences team characteristics like team size, team dynamics [10] team learning and coordination [8], however research to date has focused mainly on describing observed changes or posing future research questions.

Aiming to contribute towards increasing our understanding of the role of GenAI in agile teams, in this study we explore the following research question: *How does individual human-AI collaboration disrupt agile teams?*

Our findings show that individual human-AI collaboration disrupts agile team interaction, collaboration and learning in both positive and negative ways. For instance, individual learning strategies, when shared, could enhance team learning. However, we also find evidence of AI use-related discomfort, which we term “AI shame” and perceived problems with “opting out” for AI-skeptical team members which can negatively impact team dynamics. Such findings suggest the need to reconsider our understanding of agile teams, and teams in general [13].

2 Research Methods

Data Collection. To address our research question, we conducted thirteen interviews with software and technology professionals in Knowit, a Scandinavian consultancy firm specializing in agile software development and IT management. Interviews were conducted between June and October 2025, either face-to-face or via Microsoft Teams. Interviews lasted between 45–60 min and recorded based on participant consent. We used an interview guide structured into themes such as teamwork and collaboration, individual work, and work practices. Example questions include “*How does AI tools influence team dynamics such as collabo-*

ration and communication” and “*How do you involve AI tools in your problem solving practices?*”

Participants included six developers, two marketers, two recruiters, one AI engineer, one technology advisor, one technical architect. We strived for a varied sample to represent a broad range of perspectives and experiences, but common for all roles was that they worked in teams. Participant seniority level ranged from junior to mid-level, senior and expert, with company tenures ranging from less than a year to eight years. In terms of gender, only two participants were female (one developer and one marketer).

All data was stored in an SQLite database with anonymized identifiers which served as the central repository for codes, themes, and memos. To ensure transparency, analytical scripts are available at [link].¹

Analytical Procedures. We followed Braun and Clarke’s six-phase thematic analysis framework [4], supported by GenAI tools in our coding process, following SE best practices [11]:

1) *Getting to know the material*: Interviews were transcribed using a Whisper-based transcription tool. All transcripts were read and quality checked by members of the research team. During this procedure, we first 2) *generated initial codes* and 3) *searched for themes* manually, coding transcripts in Microsoft Word and transferring codes to our database via script. Findings were iteratively discussed within the research team. In a second pass over phases two and three, we iteratively prompted LLMs (Claude, GPT-4) to suggest additional codes and thematic groupings, which we compared against our manual set. For example, an LLM suggested codes such as `ai_before_colleague` when analyzing passages about developers preferring AI over colleagues. We critically reviewed each suggestion, accepting, modifying, or rejecting based on our domain expertise. All final decisions were made by human researchers.

Following this, the themes from the LLM-generated analysis were thoroughly 4) *reviewed*, both by having the LLM and human researchers cross-check findings against each interview transcript. Codes were grouped into working themes in the database, and we merged working themes into the six themes reported here when they covered the same idea and were raised by several participants; disagreements between the manual and LLM passes were resolved through team discussion, with all final decisions made by the human researchers. We continued this process until saturation was reached, i.e., no new data added new insights or no new themes emerged, as deemed by the human researchers. When we were satisfied with the set of themes we 5) *assigned names and descriptions to each theme* in parallel with 6) *writing the report*. No LLM support was used for analytical decisions during these final phases, however we used GenAI tools to assist writing, including formulating arguments and refining prose.

¹ Link anonymized for review; will be made public upon acceptance.

3 Findings

Our thematic analysis identified six themes: (T1) AI use is individual, not shared; (T2) AI changes team mentoring patterns (T3) Socratic prompting can enhance learning; (T4) AI changes team interaction patterns; (T5) AI-induced discomfort; and (T6) human oversight remains essential. Support varied across themes: T6 was the only one raised by all 13 participants, T1–T4 by 10–12, and T5 explicitly by a minority (5 of 13).

3.1 AI Disrupts Learning in Agile Teams

A first set of findings relates to how individual human-AI collaboration disrupted team knowledge flow. Across most participants (11 of 13), AI was predominantly used individually rather than collectively (**T1**). We found no established team-level AI guidelines. This individual adoption pattern drives the disruptions described in the following. We found that traditional mentoring patterns changed as GenAI enabled individuals to operate beyond their specialization, reducing dependency on teammates with complementary skills (**T2**). For instance, a junior AI engineer described how s/he now works across domains: *“I don’t know Javascript or frontend at all. But I have gotten requests to fix things in [the internal portal]... where I got AI to help me implement”* [P02].

However, the ability to get quick answers from AI raised concerns about long-term competence development. A junior developer explained: *“My biggest reason for skepticism is that I’m very nervous we’ll get a large gap between senior developers and juniors... We’ll get a whole generation of developers who never learn how to solve these problems by walking the path”* [P02].

Another participant reflected on how it was expertise that enabled effective AI use, not the reverse: *“In the area where I actually have competence... I experience the tool as very good. While in the area of legal language, where I feel I don’t have great competence, I notice that I don’t dare trust the tool”* [P05].

Some participants deliberately used AI to enhance rather than bypass learning (**T3**). One developer used a Socratic approach where: *“I usually system-prompt my agents to use the Socratic method. ‘Don’t give me the answer right away... give me the approach for how I get to the problem on my own’”* [P02].

Finally, participants described how the tension between productivity and learning had changed their team interaction patterns: This becomes *“a kind of ‘elephant in the room’ in every single meeting. Because you can sit and discuss... then we throw it to an AI, because we know that probably it’s good enough”* [P02].

3.2 AI Disrupts Agile Team Communication and Collaboration

A second set of themes relates to how individual AI adoption fundamentally changes how, when, and why team members interact (**T4**).

Participants consistently reported asking AI before colleagues: *“It definitely changed how I worked. I spent more time trying to talk to the AI before I checked in [with colleagues]. For things directly related to code, if I know someone else*

knows shell script, before I would have asked them directly. Now I ask GPT" [P04]. A junior developer observed the threshold for human interaction rising: *"I think it becomes a higher threshold for sitting down with each other and helping. If you get a question, it's like: 'Have you tried GPT?'"* [P09].

Not everyone, however, embraces this shift. One participant deliberately limited their AI use due to environmental and privacy concerns, yet found themselves *"forced to use AI when I don't want to"* [P08] through AI features built into the tools they use. Another who rejected AI tools *"has essentially placed himself on the sidelines, and doesn't appear as relevant anymore"* [P05].

Our data also indicates that individual human-AI collaboration has the potential to reduce teams to minimal sizes. An expert consultant described working almost entirely solo: *"I'm almost never part of a team. If I can't divide the problem... If I'm precise enough, it takes me longer to write a specification [for someone else] than to just solve it myself [using AI]"* [P06].

AI also reshapes traditional collaborative practices like pair programming which according to our data, may challenge its value: *"Sometimes you suddenly see them just starting to do a lot and you're really just sitting there watching... one person says 'that looks good'"* [P08]. This suggests how the active engagement of human pairing may be lost when AI takes over code production.

Finally, several participants described what they called "AI shame", a discomfort about relying on AI (T5). A marketing manager explained: *"I feel like I'm cheating a bit... If I present market strategy that came through ChatGPT, you didn't need me and my knowledge"* [P12]. Another admitted: *"It feels a bit like we're cheating when things are made so fast. So we... haven't been completely open about all our ChatGPT use"* [P05]. This shadow use may affect team trust and shared understanding of how work is actually done.

3.3 Human Team Members Remain Essential

Despite AI's capabilities and their implications for individual work processes, all participants emphasized that human oversight could not be delegated (T6).

A developer recounted security failures: *"It wrote something completely insecure. Something first-year students could get past... if you don't know what you're doing, it can go wrong"* [P03]. Participants further distinguished between production and decision-making: *"Often it's pure content production, presentation-making. Because as soon as things affect or touch people in any way... then I think you must sit down and actually talk to someone"* [P10]. AI handles production; humans retain responsibility for decisions affecting people.

The oversight requirement stems from fundamental AI limitations: *"Things stopped working very quickly. [...] It created a lot, but it had no idea what it created. And it didn't document anything itself"* [P01]. Another participant drew a firm boundary: *"Don't let AI maintain AI. That will happen too, but we should have more control... somewhere you have to draw a line"* [P01].

Table 1. How individual human-AI collaboration disrupts agile team practices

From (Traditional teams)	To (AI-influenced teams)
<i>Learning</i>	
Asking colleagues for help	Asking AI first
Seniors as mentors	AI as Socratic tutor
Team learning discussions	Efficiency over learning
<i>Collaboration</i>	
Shared tool decisions	Individual AI adoption
Pair/mob programming	AI-assisted programming
Interrupting colleagues	AI reduces interruptions
Transparent tool use	Hidden AI use
Cross-functional teams	Smaller generalist units
<i>Oversight</i>	
Writing code	Reviewing AI code

These findings illustrate the retained need for team accountability. While AI can take over functions previously performed by colleagues, it cannot assume the accompanying responsibilities. A senior recruiter framed it explicitly: “*I use generative AI more as a junior assistant than an equal collaborator.*” [P11].

4 Discussion and Concluding Remarks

Our findings point to a number of ways in which individual AI adoption disrupts team dynamics in agile teams, summarized in Table 1.

4.1 Implications for Theory

In line with existing studies [8,14], our findings show that AI use is still on the level of individual Human-AI interaction. We found little evidence of participants using AI actively at the team level to collaborate, a finding that is reflected in another recent study on agile practitioners [12]. However, our findings points to several areas through which cognition becomes individualized again through AI, which may have implications for e.g., team cognition, shared mental models and team trust, all of which are key characteristics of agile teams [5,9].

First, mentoring and knowledge transfer are key aspects of agile teams [9]. However, having AI assistance readily available may stifle team discussions and asking colleagues for help [3]. In line with [15], our findings suggest that while AI may support individual learning, it disrupts the collective learning that emerges from working alongside experienced colleagues. On the other hand, reduced reliance on colleagues may enable more balanced participation, particularly in teams previously dominated by senior experts, though this remains speculative.

Our findings further illustrate a competence paradox, where AI enables juniors to appear more competent (operating across domains), while potentially undermining the path to becoming actually competent [15]. Interestingly, our findings show how some individuals used a 'Socratic prompting approach' which, if used systematically as a training tool at the team level, may enforce and strengthen team learning. However, this approach must be taught. Accordingly, agile team mentoring still appears key.

Second, our findings point to a need to reconsider what an agile team should look like and do. In line with existing studies [8, 14], our findings show that AI use in our case organization is still on the level of individual human-AI interaction. However, we observed early signs of team-level AI augmentation: one participant described how AI was brought into pair programming sessions when the pair got stuck, turning the pair into a "trio." While this shows AI moving toward team augmentation [7, 13], it raises questions about how this integration affects the dynamics and learning value of traditional pair programming, as AI-skeptical team members experienced that they could not opt out.

Third, in line with other studies [2, 14, 15] we also find indications of a redefinition of work from producing code to oversight and revision of AI output. This poses interesting questions related to for instance task and role distribution, team accountability and our core understanding of agile teams. Our findings illustrate that AI can enable individuals to perform a wide set of development tasks, taken to the extreme a "one person army" where the tool is essentially replacing the team. However, this notion ignores that the agile team's value lies in the ability to leverage cross-functional shared knowledge to deliver value beyond that of any individual with seemingly limitless tool capabilities, which is part of what makes agile teams resilient [5, 9].

Finally, AI may have negative effects on individual and team social outcomes [3, 10, 13]. Our findings point to how team members that are skeptical towards AI may be 'sidelined' and left with the impression that AI-use cannot be avoided at the expense of their personal preferences. If individual AI adoption, in the absence of team policy, becomes a de facto team norm, this may challenge the individual autonomy and working styles that agile emphasizes.

Another negative implication is the discomfort resulting from AI use in settings where one perceives that AI use will have negative social implications, which we term 'AI shame'. Notably, AI shame operates on two levels: externally, as fear of social judgment from colleagues, and internally, as a diminished sense of ownership when deliverables feel "too easy" relative to perceived effort required. Since agile teams rely on transparency and psychological safety [5, 9], teams must create environments where AI use is publicly acknowledged and personally accepted. While previous studies has begun to explore the emotional dimensions related to AI use [3], this is a nascent topic in need of further research.

4.2 Implications for Practice

Our findings holds several practical implications that may enable teams to benefit from AI without sacrificing collaboration and learning.

- 1) *Establish team guidelines for AI use.* Integrate AI guidelines into team contracts to maintain collective ownership and shared understanding, and to make AI use visible and transparent. This may prevent shadow use, increase trust, and enable teams to collectively learn effective AI practices. (*Grounded.*)
- 2) *Make space for human-human learning.* Designate AI-free problem-solving sessions and mandatory pairing time for juniors to build both technical skills and domain knowledge alongside AI-assisted productivity. (*Problem grounded; intervention not tested.*)
- 3) *Teach AI as learning tool, not shortcut.* Mentor team members in deliberate AI practices that enhance rather than replace learning, such as using AI for guidance and explanation in addition to complete solutions. (*Extrapolated from 2 of 13.*)
- 4) *Make human oversight a core team responsibility.* Treat any AI output (code, text, analysis) as a “draft from a junior assistant” that requires human domain knowledge and judgment before acceptance. (*Grounded; team framing extends our data.*)
- 5) *Reconsider team size and composition carefully.* The question should be “what team structure is needed for sustainable delivery?” not just “how few people can we get away with?” (*Speculative.*)

4.3 Limitations and Concluding Remarks

We have explored how GenAI transforms agile teams. The study has limitations: a single-case design limits external generalizability, interviews provide only a snapshot of a fast-evolving landscape, and three of the authors are full-time employees in the case organization. To mitigate the last point, those authors did not interview respondents for whom they held personal responsibility, and the research process and analyses were led by the fourth author, whose primary affiliation is academic and whose link to Knowit is as a researcher with no internal company responsibilities. We registered the study with Sikt, the Norwegian agency for shared services in education and research. Despite these drawbacks, our study illustrates a need to reconsider the role of agile teams in software engineering. As research increasingly considers the role of AI in teamwork [8,13], and AI-augmented teams may replace traditional development teams [7], our understanding of agile teams must be updated. Longitudinal studies should explore how team dynamics evolve as AI adoption matures, and future work should examine “AI shame” to uncover its implications at the individual and team level. Above all, future research should revisit and update existing models of agile teamwork to develop new frameworks for the AI age.

Acknowledgments. This work was supported by the Research Council of Norway through the Kairos project (grant no. 357147) and by the European Union’s Horizon Europe research and innovation programme through the PROSPECTS 5.0 project (grant agreement no. 101135948).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Anthony, C., Bechky, B.A., Fayard, A.L.: “Collaborating” with AI: taking a system view to explore the future of work. *Organ. Sci.* **34**(5), 1672–1694 (2023)
2. Banh, L., Holldack, F., Strobel, G.: Copiloting the future: how generative AI transforms software engineering. *Inf. Softw. Technol.* **183**, 107751 (2025)
3. Barbala, A., Ulfsnes, R., Wivestad, V.T., Stray, V.G.: Generative AI in the workplace: affective affordances and employee flourishing. In: *Proceedings of the 58th Hawaii International Conference on System Sciences*, pp. 6015–6024 (2025)
4. Braun, V., Clarke, V.: Using thematic analysis in psychology. *Qual. Res. Psychol.* **3**(2), 77–101 (2006). <https://doi.org/10.1191/1478088706qp063oa>
5. Cohen, S.G., Bailey, D.E.: What makes teams work: group effectiveness research from the shop floor to the executive suite. *J. Manag.* **23**(3), 239–290 (1997)
6. Herda, T., Pichler, V., Zhang, Z., Abrahamsson, P., Hanssen, G.K.: AI and agile software development: from frustration to success XP2025 workshop summary. In: *International Conference on Agile Software Development*, pp. 3–13. Springer Nature, Switzerland, Cham (2025)
7. Hoda, R., Dam, H., Tantithamthavorn, C., Thongtanunam, P., Storey, M.A.: Augmented agile: human-centered AI-assisted software management. *IEEE Softw.* **40**(4), 106–109 (2023)
8. Kwok, Y.T.C., Adil, M.: AI and teamwork in agile software development: a systematic mapping study. In: *International Conference on Agile Software Development*, pp. 32–40. Springer (2025)
9. Moe, N.B., Dingsøyr, T., Dybå, T.: A teamwork model for understanding an agile team: a case study of a scrum project. *Inf. Softw. Technol.* **52**(5), 480–491 (2010)
10. Nguyen-Duc, A., et al.: Generative artificial intelligence for software engineering—a research agenda. *Softw. Pract. Experience* **55**(11), 1806–1843 (2025)
11. Ornelas, T., Araújo, A.A., Araújo, J., Araújo, M., Trinkenreich, B., Kalinowski, M.: LLM-assisted thematic analysis: opportunities, limitations, and recommendations. *arXiv preprint [arXiv:2511.14528](https://arxiv.org/abs/2511.14528)* (2025)
12. Planötscher, D., Wang, X., Adil, M., Gregory, P., Larimian, S.: Human-AI collaboration in software development activities: perspectives of agile practitioners. In: *International Conference on Agile Software Development*, pp. 274–289. Springer (2026)
13. Richter, A., Schwabe, G.: “there is no ‘AI’ in ‘team’! or is there?”—towards meaningful human-AI collaboration. *Australas. J. Inf. Syst.* **29** (2025)
14. Stray, V., Brandtzæg, E.G., Wivestad, V., Barbala, A., Moe, N.B.: Developer productivity with and without GitHub copilot: a longitudinal mixed-methods case study. In: *Proceedings of the 59th Hawaii International Conference on System Sciences* (2026)
15. Ulfsnes, R., Moe, N.B., Stray, V., Skarpen, M.: Transforming software development with generative AI: empirical insights on collaboration and workflow. In: *Generative AI for Effective Software Development*, pp. 219–234. Springer (2024)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





From AI Assistants to AI-Dev-Systems: A Three-Layer Model for Human-AI Collaboration in Agile Software Development

Andreas Hinderks¹(✉)(ID), Jörg Thomaschewski²(ID), and Eva-Maria Schön^{1,2}(ID)

¹ University of Applied Sciences and Arts, Hannover, Germany
jan-andreas.hinderks@hs-hannover.de, eva-maria.schoen@hs-emden-leer.de

² University of Applied Sciences, Emden/Leer, Germany
joerg.thomaschewski@hs-emden-leer.de

Abstract. Context: For decades, integrating new capabilities into agile teams, whether UX practices or emerging technologies, has followed a consistent paradigm: teams remain stable while methods adapt. Applied to AI, this approach produces fragmented tooling, governance uncertainty, and *vibe coding*: unstructured AI use that prevents cumulative organizational capability. We argue this integration paradigm, successful for disciplines like UX, may be inadequate for AI's transformative scope.

Objective: We reframe AI integration as an organizational design problem: how should teams restructure around *AI-Dev-Systems*, configured environments of AI agents, workflows, and knowledge bases functioning as shared infrastructure?

Method: We propose a three-layer model extending platform engineering: (1) *System Teams* design AI-Dev-Systems; (2) *Platform Teams* maintain them; (3) *Product Teams* develop software within them.

Results: Developers shift from producers to stewards monitoring AI outputs. Iteration cycles compress, paradoxically elevating product thinking: when implementation ceases to be the bottleneck, *what* to build matters more than *how*. XP practices persist but transform into governance mechanisms for AI-generated work.

Conclusion: Teams should restructure around AI-Dev-Systems rather than adopt AI tools, our three-layer model offers a structured provocation to reframe the discourse.

Keywords: AI-Dev-Systems · Agile Software Development · Platform Engineering · Human-AI Collaboration

1 Introduction

The integration of AI coding assistants into software development teams raises fundamental questions about organizational design. As these tools become

increasingly capable, organizations face decisions that extend beyond tool selection: they must consider how AI-augmented development affects team structures, individual roles, and collaborative workflows. This is not merely a technical challenge but an organizational one.

Current approaches typically treat AI integration as a tool adoption problem, asking how to fit AI assistants into existing team structures. This framing, however, may underestimate the organizational implications. At the XP 2025 workshop, 73.3% of participants identified fragmented tooling as their primary frustration with AI-assisted development [16]. When each developer independently adopts and configures AI tools, the result is often a proliferation of incompatible setups rather than coherent team infrastructure.

This challenge echoes earlier efforts to integrate new disciplines into agile teams. In 2023, we explored a similar question in the context of User Experience, guest-editing a special issue on Agile UX integration [12]. The central question then was how to embed UX practices into existing agile team structures, a paradigm that assumed teams remain stable while new capabilities adapt to fit them. That approach proved largely successful: UX practices could adapt to agile rhythms without requiring fundamental team restructuring. We already noted AI tools as an emerging force, but underestimated their organizational implications.

AI, however, operates differently. AI-Dev-Systems, configured environments comprising AI agents, automated workflows, and curated knowledge bases, are not merely tools that adapt to teams; they are infrastructure that fundamentally shapes how development work can be performed. This suggests an inversion: rather than asking how teams can accommodate AI assistants, we should examine how organizations might restructure around AI-Dev-Systems. This perspective draws on sociotechnical systems theory, which emphasizes that technological change requires corresponding attention to institutions and governance [8], and on platform engineering principles that demonstrate how well-designed internal platforms reduce cognitive load while enabling team autonomy [13].

This position paper introduces a three-layer model distinguishing teams that design, maintain, and operate AI-Dev-Systems. Section 2 examines current approaches and their limitations. Section 3 presents our organizational model. Section 4 explores implications for agile practices and poses open questions for workshop discussion.

2 Background: The AI-as-Assistant Paradigm

The dominant approach to AI in agile software development treats AI as an assistant integrated into existing team structures. Research explores how LLM-powered meeting assistants support Daily Scrums [2], how code copilots accelerate individual developer tasks [10], and how chatbots help refine user stories [16]. Industry studies report productivity gains of 20–50% for bounded tasks such as unit test generation and boilerplate coding [3, 5]. The underlying assumption is consistent: team structures remain stable while AI tools adapt to existing workflows. This framing positions AI integration as a tool adoption problem.

This assumption encounters systematic difficulties at the team level. The XP2025 workshop synthesized over 120 practitioner pain points, with fragmented tooling emerging as the most frequently cited frustration (73.3% of participants) [7]. Practitioners described a “paradox of choice” in which overlapping, non-interoperable AI tools create cognitive overhead rather than reduce it [16]. Governance uncertainty compounds the problem: teams lack clarity on data privacy boundaries when AI models ingest sprint artifacts, leading to adoption delays or abandonment [16]. Skills gaps further limit systematic adoption, prompt engineering literacy remains rare, and early abandonment after unsuccessful initial trials is common [7].

The consequence is what we term *vibe coding*: unstructured, opportunistic AI use that produces isolated successes but fails to build cumulative organizational capability. This pattern explains a puzzling finding in recent empirical work: developers report high satisfaction with AI tools even as teams struggle to integrate them into coherent workflows [15]. Individual task productivity may rise while team collaboration and knowledge sharing decline [14]. The assistant paradigm optimizes for ad-hoc individual adoption rather than organizational infrastructure.

We argue this reflects a category error. AI integration is not primarily a tool adoption problem; it is an organizational design problem. Sociotechnical systems theory holds that successful technology integration requires corresponding attention to institutions, culture, and governance, not just technical artifacts [8]. Treating AI as “just another tool” neglects the structural changes required to leverage it effectively. The productive question is not how to add AI assistants to existing teams, but how to design teams and infrastructures that make AI systematically useful. The following section proposes such a design.

3 The Three-Layer Model

Platform engineering offers a proven response to tool proliferation: abstract complexity into stable platforms that *Product Teams* consume as self-service capabilities [13]. We propose that AI-augmented development requires an analogous structural response, not adding AI tools to existing teams, but designing teams around shared AI infrastructure.

We define an *AI-Dev-System* as a configured environment of AI agents, tools, workflows, and knowledge bases designed for systematic software development. Examples range from custom configurations, Claude with Model Context Protocol integrations, Cursor with project-specific rules, to purpose-built environments such as Design OS and Agent OS [1]. The distinguishing feature is intentional design: deliberate choices about agent capabilities, interaction patterns, and domain knowledge that apply across a team.

Research on LLM-based multi-agent systems supports this framing. He et al. [6] identify orchestration platforms as core infrastructure managing coordination among agents. Yet this organizational question often remains implicit: who builds and evolves these systems? Our three-layer model makes this division of labor explicit (Fig. 1).

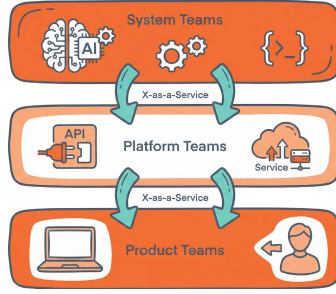


Fig. 1. The three-layer model: *System Teams* design, *Platform Teams* maintain, and *Product Teams* operate within the configured environment. Arrows indicate capability provision (downward) and feedback flow (upward).

3.1 Layer 1: System Teams

System Teams design and construct AI-Dev-Systems. Their responsibilities include agent configuration, workflow design, prompt engineering, knowledge base curation, and integration architecture.

In Team Topologies terminology [13], *System Teams* function as a *complicated subsystem team*: specialists who shield other teams from orchestration complexity. Crucially, this expertise is concentrated: not every developer needs deep prompt engineering skills if architects encode that knowledge into system design.

3.2 Layer 2: Platform Teams

Platform Teams maintain and evolve the AI-Dev-System after deployment. Their responsibilities include bug fixes, model updates, capability extensions, and regression management. This mirrors conventional platform engineering [13], but AI infrastructure demands higher operational velocity.

Two factors drive this tempo. First, foundation model updates occur frequently and can alter system behavior unpredictably. Second, *Product Teams* generate continuous feedback about workflow friction; *Platform Teams* translate this into improvements. The feedback loop between Layers 2 and 3 determines whether the AI-Dev-System accumulates organizational learning or stagnates.

3.3 Layer 3: Product Teams

Product Teams develop software *using* the AI-Dev-System as their primary environment. Rather than selecting AI tools individually, they consume curated capabilities provided by the platform. This addresses the “paradox of choice” identified in Sect. 2.

This restructures the developer role. Following He et al. [6], developers increasingly function as *monitoring developers*: observing AI-generated outputs,

validating results, and steering the development process. Traditional XP practices, pair programming, test-driven development, collective ownership, remain valuable, but shift from direct production methods to governance mechanisms for AI-generated work.

4 Implications and Open Questions

If teams restructure around AI-Dev-Systems, what changes for agile practices, and what remains?

4.1 What Changes

From Producer to Steward. The most significant shift concerns the developer's role. As discussed in Sect. 3.3, developers function as monitors: observing outputs, validating results, and intervening at decision points. The task shifts from producing code to stewarding production.

Human-human pairing involves continuous dialogue and shared reasoning. Human-AI collaboration differs: the AI produces, the human evaluates and redirects. Research indicates this requires distinct competencies, particularly calibrating trust appropriately [9, 11]. Pair programming skills do not transfer automatically.

Compressed Iteration Cycles. When AI-Dev-Systems generate prototypes in hours, the two-week sprint becomes coarse-grained. Teams plan around system capabilities and validation checkpoints rather than individual task assignments. The sprint may persist as coordination rhythm while losing its function as production cadence.

System-Level Retrospectives. Retrospectives expand to include the AI-Dev-System itself. *Product Teams* evaluate the system as an upstream dependency, surfacing friction points. This feedback flows to *Platform Teams*, making retrospectives a governance mechanism for system evolution.

4.2 What Remains

The three-layer model recontextualizes agile values rather than abandoning them. Iterative development remains essential: AI-Dev-Systems do not eliminate the need for rapid feedback, customer collaboration, or responsiveness to change. Compressed cycles intensify rather than diminish these requirements.

XP practices retain relevance but shift in application:

Test-driven development becomes a specification mechanism for AI-generated code. Tests declare intent; AI outputs must satisfy them.

Collective ownership extends to the AI-Dev-System itself. *Product Teams* share responsibility for validating system outputs and providing improvement feedback.

Continuous integration ensures AI-generated contributions meet quality standards before merging, with automated checks compensating for non-deterministic production.

Human judgment remains irreplaceable for architecture decisions, ethical considerations, stakeholder communication, and contextual reasoning beyond current AI capabilities [6]. The three-layer model structures these responsibilities rather than automating them.

4.3 Open Questions

We offer the following questions to structure workshop discussion.

- *Q1: Competency Boundaries.* What distinguishes *System Teams* from Product Developers? What training prepares someone to construct agent configurations and design workflows? Is this a new specialization or an evolution of existing roles?
- *Q2: Governance Without Rigidity.* How do organizations govern AI-Dev-Systems while preserving team autonomy? Platform engineering succeeds partly because platforms provide capabilities without dictating usage [13]. Can this balance hold with non-deterministic agents?
- *Q3: Trust Dynamics.* Does operating within a curated AI-Dev-System improve trust calibration by providing consistent capabilities [9]? Or does abstraction create over-reliance risks, trusting the platform without understanding its limitations?
- *Q4: Scaling and Collapse.* Does the model scale in both directions? Small teams cannot support dedicated roles. Can it collapse gracefully? Do large organizations require additional layers?
- *Q5: Effectiveness Metrics.* How should organizations measure AI-Dev-System value? Task-level metrics miss system-level outcomes. Developer experience research suggests flow state, cognitive load, and feedback quality matter [4], how do these apply here?

4.4 Propositions

We close with three deliberate propositions intended to challenge assumptions.

Propositions 1: The Assistant Metaphor is Harmful. Framing AI as an “assistant” or “copilot” implies a peer relationship that obscures the infrastructure character of AI-Dev-Systems. Assistants help; infrastructure enables. The metaphor may be actively impeding organizational thinking about AI integration.

Propositions 2: Programming Becomes Secondary. If *Product Teams* primarily steer AI agents, the core developer competency shifts from writing code to directing its production. Programming literacy remains necessary, one cannot evaluate what one does not understand, but it becomes a prerequisite rather

than the primary activity. The developer role evolves toward editor or director: shaping outputs rather than creating them.

Propositions 3: Agile Teams May Not Survive AI-Dev-Systems. The three-layer model preserves agile values but restructures agile teams. If *System Teams* and *Platform Teams* absorb complexity that previously resided in cross-functional *Product Teams*, the self-organizing team, a core agile principle, may fragment into specialized layers. Is this acceptable? Or does it represent a fundamental departure from agile philosophy that we should resist?

These propositions are intended to challenge, not assert. We invite workshop participants to refute, refine, or extend them.

5 Conclusion

The dominant question in AI-augmented software development, how do we integrate AI assistants into our teams?, assumes teams are fixed while AI adapts. We have argued for an inversion: teams should restructure to operate within AI-Dev-Systems, not merely adopt AI tools. This reframing shifts the unit of analysis from tool adoption to system design.

Our three-layer model, *System Teams* who design AI-Dev-Systems, *Platform Teams* who maintain and evolve them, and *Product Teams* who develop software within them, extends established platform engineering patterns to AI-augmented development. The model is not definitive; it is a structured provocation intended to open design space that the current discourse forecloses.

This position paper is a conversation starter, not a conclusion. We invite workshop participants to challenge our premises, refine our model, or propose alternatives. The questions we have raised, about skills, governance, trust, scaling, and metrics, require collaborative investigation across research and practice. If we have succeeded, it is in demonstrating that how we frame the AI-agile question determines what solutions become visible.

Acknowledgments. We note that text-parts of this paper were refined with the support of the GenAI tools Claude (Opus 4.5).

References

1. Builder Methods: Design OS and agent OS: AI-powered development systems (2025). <https://buildermethods.com>. Accessed 15 Jan 2025
2. Cabrero-Daniel, B., Herda, T., Pichler, V., Eder, M.: Exploring human-AI collaboration in agile: customised LLM meeting assistants. In: Agile Processes in Software Engineering and Extreme Programming. LNBIP, vol. 512, pp. 163–178. Springer (2024). https://doi.org/10.1007/978-3-031-61309-9_11
3. Chui, M., Hazan, E., Roberts, R., Singla, A., Smaje, K.: The economic potential of generative AI: the next productivity frontier. Technical report, McKinsey & Company (2023). <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier>

4. Forsgren, N., Storey, M.A., Maddila, C., Zimmermann, T., Houck, B., Butler, J.: DevEx: what actually drives productivity. *ACM Queue* **21**(2), 35–53 (2023). <https://doi.org/10.1145/3595878>
5. GitHub: Octoverse: The state of open source and rise of AI in 2023. Technical report, GitHub (2024). <https://github.blog/news-insights/research/the-state-of-open-source-and-ai/>
6. He, J., Treude, C., Lo, D.: LLM-based multi-agent systems for software engineering: literature review, vision, and the road ahead. *ACM Trans. Softw. Eng. Methodol.* **34**(5), 1–30 (2025). <https://doi.org/10.1145/3712003>
7. Herda, T., Pichler, V., Zhang, Z., Abrahamsson, P., Hanssen, G.K.: AI and agile software development: from frustration to success - XP2025 workshop summary. In: *Agile Processes in Software Engineering and Extreme Programming - Workshops. LNBP*, vol. 561, pp. 3–13. Springer (2025). https://doi.org/10.1007/978-3-032-05799-0_1
8. Kudina, O., van de Poel, I.: A sociotechnical system perspective on AI. *Minds Mach.* **34**(21), 1–9 (2024). <https://doi.org/10.1007/s11023-024-09676-8>
9. Mozannar, H., et al.: Investigating and designing for trust in AI-powered code generation tools. In: *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency, FAccT 2024*, pp. 1–12. ACM (2024). <https://doi.org/10.1145/3630106.3658984>
10. Peng, S., Kalliamvakou, E., Cihon, P., Demirer, M.: The impact of AI on developer productivity: evidence from GitHub Copilot (2023). arXiv preprint [arXiv:2302.06590](https://arxiv.org/abs/2302.06590)
11. Schelble, B.G., Flathmann, C., McNeese, N.J., Ragan, E.D.: Shaping a multidisciplinary understanding of team trust in human-AI teams. *Eur. J. Work Organ. Psy.* **33**(2), 171–188 (2024). <https://doi.org/10.1080/1359432X.2023.2200172>
12. Schön, E.M., Silva da Silva, T., Hinderks, A., Sharp, H., Thomaschewski, J.: Introduction to special issue on agile UX: challenges, successes and barriers to improvement. *Inf. Softw. Technol.* **158**, 107193 (2023). <https://doi.org/10.1016/j.infsof.2023.107193>. <https://www.sciencedirect.com/science/article/pii/S0950584923000472>
13. Skelton, M., Pais, M.: *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. IT Revolution Press (2019)
14. Ulfesnes, R., Moe, N.B., Stray, V., Skarpen, M.: Transforming software development with generative AI: empirical insights on collaboration and workflow. In: *Generative AI for Effective Software Development*, pp. 219–234. Springer Nature, Switzerland (2024). https://doi.org/10.1007/978-3-031-55642-5_10
15. Wivestad, V.T., Barbala, A., Stray, V.: Copilot’s island of joy: balancing individual satisfaction with team interaction in agile development. In: *Agile Processes in Software Engineering and Extreme Programming. LNBP*, pp. 123–129. Springer Nature, Switzerland (2024) https://doi.org/10.1007/978-3-031-61309-9_8
16. Zhang, Z., Herda, T., Pichler, V., Abrahamsson, P., Hanssen, G.K.: AI and agile software development: a research roadmap from the XP2025 workshop (2025). arXiv preprint [arXiv:2507.20563](https://arxiv.org/abs/2507.20563)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Human–AI Collaboration for Scaling Agile Regression Testing: An Agentic-AI Teammate from Manual to Automated Testing

Moustapha El Outmani¹✉, Manthan Venkataramana Shenoy¹,
Ahmad Hatahet¹, Andreas Rausch¹, Tim Niklas Kniep²,
Thomas Raddatz², and Benjamin King²

¹ Institute for Software and Systems Engineering, Technical University of Clausthal,
Adolph-Roemer-Straße 2A, 38678 Clausthal-Zellerfeld, Germany
{moustapha.outmani,manthan.shenoy,ahmad.hatahet,
andreas.rausch}@tu-clausthal.de

² Hacon Ingenieurgesellschaft mbH – A Siemens Company, Lister Str. 15, 30163
Hannover, Germany
info@Hacon.de
<https://www.isse.tu-clausthal.de/> , <https://www.Hacon.de/>

Abstract. Automated regression testing is essential for maintaining rapid, high-quality delivery in Agile and Scrum organizations. Many teams, including Hacon (a Siemens company), face a persistent gap: validated test specifications accumulate faster than they are automated, limiting regression coverage and increasing manual work. This paper reports an exploratory industrial case study of the Hacon Test Automation Copilot, an agentic AI system that generates system-level regression test scripts from validated specifications using retrieval-augmented generation and a multi-agent workflow. Integrated with Hacon’s CI pipelines, the Copilot operates asynchronously as a “*silent AI teammate*”, producing candidate scripts for human review. Mixed-method evaluation shows the AI accelerates script authoring and increases throughput, with 30–50% code reuse. However, human review remains necessary for maintainability and correct domain interpretation. Clear specifications, explicit governance, and ongoing human–AI collaboration are critical. We conclude with lessons for scaling regression automation and enabling effective human–AI teaming in Agile settings.

Keywords: Agile · Scrum · Regression Testing · Test Automation · Human–AI Collaboration · Agentic AI · Continuous Integration

1 Introduction

Agile software teams rely on continuous integration and automated regression testing to maintain quality while delivering features at speed [4,5,9]. Yet, in

large-scale environments such as Hacon (a Siemens company), the pace of evolving requirements and ongoing feature development outstrips the capacity to automate regression tests, resulting in a persistent gap between the volume of manual test specifications and the rate at which they are converted into executable scripts. This gap not only limits regression coverage and leads to increased manual testing effort in sprint releases, challenging the core agile principles of rapid feedback and sustainable delivery.

While recent advances in large language models (LLMs) and agentic AI offer promising capabilities for code and test generation [1], there is limited industry guidance on integrating such tools into agile test engineering. Existing research often overlooks the nuanced requirements of regression test suites and rarely addresses the human factors of collaboration, review, and governance in real-world teams [2, 6, 7, 10].

This paper presents an exploratory industrial case study with Hacon’s agile test engineering team, examining the introduction of an agentic AI system for automated test script generation and its integration into the team’s workflow. We address the following research question:

RQ: How does integrating an agentic AI system for automated test generation affect productivity, test script quality, and human–AI collaboration within an agile testing team?

We introduce an agentic AI teammate that generates executable system test scripts from validated specifications using retrieval-augmented generation (RAG) and a bounded multi-agent workflow, integrated with Hacon’s CI pipelines and supporting human review and artifact governance. Evaluation is based on artifact analysis and practitioner feedback.

Our contributions are: (*C1*) We present the design of an agentic AI system that transforms validated manual test specifications into executable regression test scripts within an agile testing workflow. (*C2*) We describe governance, collaboration, and review practices for integrating an AI teammate into agile test engineering workflows. (*C3*) We empirically evaluate the impact of this approach on productivity, test script quality, and human–AI collaboration through an industrial case study.

The rest of the paper is structured as follows: Sect. 2 describes the industrial context and the manual-to-automated testing bottleneck at Hacon. Section 3 presents the architecture of the Hacon Test Automation Copilot. Section 4 explains how the Copilot is integrated into the agile testing workflow and how human–AI collaboration is organized. Section 5 reports the empirical evaluation and key findings from the industrial case study. Section 6 discusses implications, limitations, and directions for future work.

2 Industrial Context and Problem Statement

Large industrial software organizations developing complex, long-lived systems increasingly rely on automated regression testing to manage scale and frequent

change [3,8,11]. Hacon (a Siemens company) represents such an industrial setting, developing large-scale transportation planning and scheduling software within a mature agile development organization, delivering frequent releases through short development sprints supported by robust CI pipelines and established system-level test automation. In this environment, regression testing involves repeatedly executing system-level tests to ensure that newly introduced changes do not break existing functionality. These tests are continuously integrated and executed within the shared CI pipeline, with the studied test engineering team collaborating closely with development teams throughout the agile development process.

Within Hacon’s testing workflow, test specifications are first created and validated by test engineers before being manually translated into executable automation scripts. This translation step requires significant engineering effort and domain expertise, resulting in a bottleneck where the number of validated test specifications grows faster than the team’s capacity to automate them.

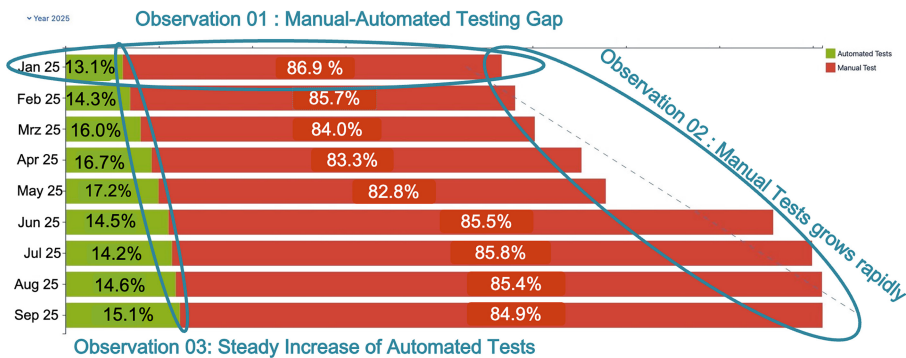


Fig. 1. Growth of manual and automated system tests across monthly releases in 2025. Manual tests dominate the regression suite, while automated coverage increases gradually, illustrating the persistent gap between manual specifications and automated scripts.

Despite continuous investment in automation and dedicated test engineering teams, Hacon faces a persistent challenge: manual test specifications continue to outpace the rate at which they are automated. As shown in Fig. 1, automated test coverage increases steadily by 1–2% per release, yet manual tests still account for 82–87% of the total, with their absolute number growing by 10–20% per release. This ongoing automation gap results in increased manual testing effort in the regression suite and slower feedback cycles, motivating the need for AI-assisted approaches that can accelerate script creation while upholding Hacon’s high standards for quality, maintainability, and governance.

3 System Architecture: Hacon Test Automation Copilot

To address the automation bottleneck described in Sect. 2, we developed the *Hacon Test Automation Copilot*, an agentic AI system that generates executable system-level test scripts from validated specifications. The Copilot operates as an asynchronous, batch-oriented collaborator, preparing candidate automation artifacts ahead of sprint activities and integrating with Hacon’s continuous integration (CI) workflow.

Test specifications are placed in a designated input folder and processed via a command-line trigger. As shown in Fig. 2, the Copilot processes each specification and produces review-ready artifacts: the generated script, Jenkins execution logs, structured summaries, and execution traces in MLflow. Test engineers review these outputs to decide on acceptance, refactoring, or rewriting before adding scripts to the regression suite.

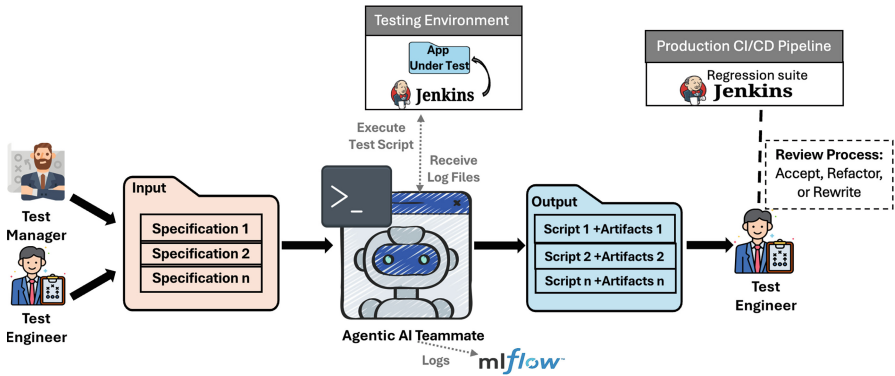


Fig. 2. Overview of the Hacon Test Automation Copilot architecture and its interaction with test engineers. The system generates candidate test scripts from validated specifications, executes them in the CI environment, evaluates the results, and produces review-ready artifacts for human validation before integration into the regression suite.

The Copilot’s architecture implements a bounded generation–execution–evaluation loop, orchestrated to respect iteration limits and CI resource availability. Within each cycle, a **Generator agent** uses retrieval-augmented generation (RAG) over historical specification–script pairs to create candidate test scripts. These are executed in a Jenkins environment, with the application under test and logs collected. An **Evaluator agent** analyzes execution logs for syntactical and semantic correctness, coverage, and improvement potential. A **Reporter agent** consolidates results into structured summaries and execution reports for both test managers and engineers. All traces are recorded in MLflow for traceability.

By operating asynchronously and **autonomously**, the agentic AI teammate generates candidate automation artifacts in advance, allowing engineers

to review them without interrupting ongoing development activities. The system is configurable and supports maintainability by expanding the RAG base and updating prompts. The architecture is designed for seamless integration and future extensibility.

4 Integration and Human–AI Collaboration in the Agile Testing Workflow

The Copilot was integrated into Hacon’s agile testing workflow as a *silent AI teammate*, operating asynchronously to generate candidate automation scripts before each sprint. This fits the team’s sprint-based process, where regression tests must be ready and validated quickly. Instead of continuous interaction, the Copilot produces first-draft artifacts for later review and refinement by test engineers prior to inclusion in the regression suite.

Test specifications, created and validated by engineers and managers, are exported as input to the Copilot. The AI system generates candidate scripts, which the test automation team reviews at the sprint’s start. This setup lets the AI handle repetitive script generation and initial validation, while engineers focus on interpretation, refinement, and approval. Engineers review artifacts and execution evidence to ensure tests match the intended behavior. Scripts are modified or rewritten as needed before acceptance.

Governance is ensured by explicit human oversight and artifact traceability. The Copilot has *bounded autonomy*: it cannot add scripts to the regression suite without human approval, and all outputs are recorded. This review process keeps automated test generation aligned with quality standards and maintains accountability.

5 Evaluation and Results

This section presents the empirical evaluation of the agentic AI system’s impact on productivity, test script quality, and human–AI collaboration in Hacon’s agile test engineering workflow.

Study Design and Data Collection

The evaluation was conducted over a four-week period with Hacon’s test engineering team (five engineers). We selected 61 test specifications (Xray format), spanning six functional areas and varying in complexity (2–18 steps per case, story point estimates 3–8, and pre-rated input clarity from A to D). For each, the Copilot generated an initial test script, which was then reviewed and refactored by a human engineer. Data collection included:

- **Structured Survey:** Engineers rated each script using a standardized form, assessing block-level correctness, overall quality, readability, usability, and test smells (1–5 Likert scale), and provided open-text recommendations. The survey also covered the clarity, completeness, accuracy, and actionability of the generated markdown reports.

- **Artifact Comparison:** For a focused subset of approximately 20 representative cases, two expert test engineers conducted a detailed artifact comparison. Using version control, they systematically compared the AI-generated scripts to the final, human-corrected versions, assessing semantic equivalence, maintainability, and recurring issues.

Key Results and Insights

1. Productivity. Artifact comparison revealed that, on average, 30–50% of the AI-generated code per script was retained unchanged by test engineers. This indicates that the AI teammate can provide a substantial head start in script authoring, reducing the time required to move from specification to executable test.

2. Test Script Quality. Most AI-generated scripts needed moderate to major human edits: of 49 scripts, 15 were fully rewritten, 20 required major, 13 medium, and 1 minor changes. Common issues included: ‘...test data should be externalized...’, ‘...unnecessary imports...’. Typical issues were hardcoded data (31/49), redundant imports (29/49), unused objects (23/49), and missing or incorrect validations (over 30 cases). These issues primarily affected maintainability, readability, and long-term usability rather than basic functional correctness. Such quality concerns are critical in Agile and Scrum environments, where regression suites must remain stable, maintainable, and easy to evolve alongside frequent releases.

3. Human-AI Collaboration. Version control analysis of all corrected scripts, together with qualitative feedback, highlighted a dynamic collaboration pattern. Engineers frequently replaced AI-generated code blocks with established, trusted snippets from previous projects, even when the AI’s logic was technically correct. This behavior reflected a preference for familiar and maintainable code. The artifact comparison further showed that the AI’s literal interpretation of specifications—stemming from the fact that test specs were designed for humans who intuitively know what to ignore or adapt. Human review remained essential, especially to address implicit domain knowledge and ensure production readiness.

6 Discussion, Lessons Learned and Future Work

Discussion. This exploratory industrial case study shows that integrating an agentic AI teammate into an Agile regression testing workflow can accelerate the conversion of validated test specifications into automated scripts, but it also exposes significant limitations. While the AI system provided a measurable productivity boost—helpful in about 30% of evaluated cases—most generated scripts required moderate to major human intervention before they could be

used in the regression suite. Client feedback highlighted that the agent often produced solutions that were “technically correct, which in this case is a bad kind of correct,” meaning the scripts met the literal requirements but failed to align with maintainability and domain-specific expectations valued by Agile and XP teams. The client also questioned the representativeness of the test set, the completeness of test descriptions, and identified the need for further evaluation and possibly explicit Framework domain convention support. These points challenge the generalizability of our findings and suggest that the effectiveness of AI-driven automation is highly dependent on the quality and clarity of input specifications, as well as on ongoing human review.

The results also surfaced key assumptions that do not always hold in real-world Scrum and Agile environments. The AI system’s performance relied on the premise that test specs are complete, data is fully known and available, and existing automation scripts are both accurate and stable—assumptions that practitioners recognized as fragile. The agent’s literal approach exposed ambiguities and gaps that human testers routinely resolve using tacit knowledge, reinforcing that human oversight remains indispensable for ensuring semantic correctness, maintainability, and trust in the regression suite. While the client’s goal is to close the automation gap with lower cost and moderate investment by “automating the automation with AI,” our findings indicate that sustainable gains require not only technical advances but also improved specification practices and explicit human–AI collaboration. Future work should focus on co-adaptation: enhancing AI capabilities through feedback and better prompts, and supporting teams in authoring clearer, more actionable test specifications.

Lessons Learned. key lessons emerged from deployment and evaluation:

- **Specification Clarity is Foundational:** AI effectiveness depends on clear, complete specifications. Human engineers routinely bridge gaps using tacit knowledge, but the AI requires unambiguous, comprehensive instructions.
- **AI Follows, Humans Lead:** The AI’s literal interpretation of specifications can lead to “technically correct” but contextually inappropriate solutions, underscoring the need for domain-informed review.
- **Human Review Is Indispensable:** Despite automation, human oversight is indispensable for ensuring semantic correctness, maintainability, and trust in the regression suite.
- **Assumptions Must Be Challenged:** The project surfaced several implicit assumptions—about test set representativeness, data availability, and maintenance effort—that require ongoing scrutiny as the system evolves.
- **Co-Adaptation is Key:** Sustainable productivity gains will require both improved AI adaptation (via better prompts, feedback loops, and semantic evaluation) and changes in human practices (such as specification authoring and review routines).

Limitations. The study is limited to one company, a small team, and a specific set of scripts. Results may not generalize to other settings. Practitioner feedback may be biased, and the evaluation period was short. Technical choices in orchestration, prompt engineering, and integration may affect reproducibility.

Future Work. Future work will focus on co-adaptation between humans and AI. We plan to (1) improve AI adaptation by integrating expert feedback and adding specialized agents for semantic evaluation, and (2) support teams in standardizing specification practices and authoring clearer test steps. These steps aim to further reduce the gap between specifications and scripts, and to strengthen human–AI collaboration for more effective and scalable regression automation.

Acknowledgments. We gratefully acknowledge the partnership and support of Hacon (a Siemens company) in providing the industrial context, data, and invaluable practitioner feedback for this study. Their collaborative engagement was essential to the system’s deployment and evaluation.

Disclosure of Interests. All authors declare that they have no conflicts of interest.

References

1. Augusto, C., Bertolino, A., De Angelis, G., Lonetti, F., Morán, J.: Large language models for software testing: a research roadmap (2025). arXiv preprint [arXiv:2509.25043](https://arxiv.org/abs/2509.25043)
2. Böckeler, B.: Understanding spec-driven-development: Kiro, spec-kit, and Tessl — martinowler.com. <https://martinfowler.com/articles/exploring-gen-ai/sdd-3-tools.html>. Accessed 21 Jan 2026
3. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.: 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>
4. Forsgren, N., Humble, J., Kim, G.: Accelerate: the science of lean software and DevOps: building and scaling high performing technology organizations. IT Revolution (2018). <https://books.google.de/books?id=Kax-DwAAQBAJ>
5. Kim, G., Behr, K., Spafford, G.: The phoenix project: a novel about IT, DevOps, and helping your business win. The Phoenix Project, IT Revolution (2018). <https://books.google.de/books?id=H6x-DwAAQBAJ>
6. Ndukwe, N.: Software testing metrics that matter: from defect density to compliance reporting in enterprises — qodo.ai. <https://www.qodo.ai/blog/software-testing-metrics/>. Accessed 21 Jan 2026
7. Neumann, M., et al.: Between policy and practice: Genai adoption in agile software development teams (2026). arXiv preprint [arXiv:2601.07051](https://arxiv.org/abs/2601.07051)
8. Rothermel, G., Harrold, M.J.: A safe, efficient regression test selection technique. *ACM Trans. Softw. Eng. Methodol.* **6**(2), 173–210 (1997). <https://doi.org/10.1145/248233.248262>
9. Schwaber, K., Beedle, M.: Agile Software Development with Scrum, 1st edn. Prentice Hall PTR, USA (2001)
10. Tawosi, V., Ramani, K., Alamir, S., Liu, X.: ALMAS: an autonomous LLM-based multi-agent software engineering framework (2025). arXiv preprint [arXiv:2510.03463](https://arxiv.org/abs/2510.03463)
11. Yoo, S., Harman, M.: Regression testing minimization, selection and prioritization: a survey. *Softw. Test. Verif. Reliab.* **22**(2), 67–120 (2012). <https://doi.org/10.1002/stv.430>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



**International Workshop on AI
for Technical Debt Management in Agile
Software Development**



AI for Technical Debt Management: Summary from the AI4TD Workshop at XP2026

Antonio Martini¹(✉) , Phu Nguyen² , Lina Ochoa³ ,
Jan Henrik Gundelsby⁴ , and Mili Orucevic⁵

¹ University of Oslo, Oslo, Norway
antonima@uio.no

² SINTEF, Oslo, Norway
phu.nguyen@sintef.no

³ Eindhoven University of Technology, Eindhoven, Netherlands
l.m.ochoa.venegas@tue.nl

⁴ Knowit, Oslo, Norway
jhg@knowit.no

⁵ Visma, Oslo, Norway
mili.orucevic@visma.com

Abstract. Modern agile development and time-to-market pressures often lead to significant hidden technical debt (TD), which must be managed for sustainable software delivery. The growing use of artificial intelligence (AI) for code generation can further increase TD if applied improperly. At the same time, software processes are becoming increasingly data-driven. This creates an opportunity: using AI to enable data-driven TD management in agile environments. This workshop offers a platform for researchers and practitioners to share ideas, experiences, and future visions on applying AI for TD management in agile development. The workshop was held in a half day and it consisted of three phases informing each other: (1) scientific talks (i.e., two paper presentations and two keynotes) that framed current challenges and their relevance; (2) two industrial talks shaping these problems in practice; and interactive sessions (i.e., group discussions and a fish bowl panel) that allowed participants to reflect about the current status of AI with respect to TD and next steps. These discussions define a research agenda that spans measurement, practices, education, and legacy modernization, and call for tight collaboration between industry and academia to keep pace with AI-augmented software development.

Keywords: agile development · technical debt · AI

1 Introduction

In 1993, Cunningham [1] used the *debt* metaphor to refer to the incurred costs of speeding up development while compromising system quality and design; we refer

© The Author(s) 2027

A. Goldman et al. (Eds.): XP 2026 Workshops, LNBIP 596, pp. 91–99, 2027.

https://doi.org/10.1007/978-3-032-39205-3_10

to these costs as technical debt (TD). In modern agile software development, the ability to obtain fast, actionable feedback on the accumulation of TD and its associated risks is paramount for developers and managers. The rapid adoption of artificial intelligence (AI) in software engineering brings a dual reality for TD: fast paced AI-generated code can introduce a vast amount of TD, while new AI-powered approaches also offer capabilities to detect, quantify, and help remediate debt. Agile teams in industry face pressure to deliver faster and incorporate AI-assisted tools into their workflow, which changes how debt accumulates, is perceived, and is managed within backlogs, sprint planning, and release pipelines.

At the same time, software processes are becoming increasingly data-driven. The growing volume and richness of data generated by these processes makes it possible to identify patterns and inefficiencies that were previously difficult to detect. This creates an opportunity: using AI to enable data-driven TD management in agile environments. The socio-technical implications of integrating AI into TD management can be vast. While it promises to reduce dependencies and bottlenecks, it also risks eroding developers' understanding of the system—by for instance, hiding crucial architectural decisions and constraints that would otherwise be learned through direct engagement [2]. Therefore, this workshop offers a platform for researchers and practitioners to share ideas, experiences, and future visions on applying AI for TD management in agile development.

The guiding theme of the AI4TD workshop is: *What implications does artificial intelligence have in the accumulation and management of technical debt in agile development?* AI4TD is the first proposed workshop dedicated to the discussion of how AI and TD are interrelated, how to exploit opportunities and highlight challenges, especially dedicated to agile software development. Being the first of its kind, and given that the intersection of AI and TD is an emerging area of research, AI4TD provided a venue for researchers and practitioners to exchange and discuss trending views, ideas, state-of-the-art, work in progress, and scientific results highlighting aspects of AI and TD.

The workshop was held in a half day and it consisted of three phases informing each other: (1) scientific talks (i.e., paper presentations and keynotes) that framed current challenges and their relevance (cf. Sects. 2 and 3); (2) industrial talks shaping these problems in practice (cf. Sect. 4); and (3) interactive sessions (i.e., group discussions and fish bowl) that allowed participants to reflect about the current status of AI with respect to TD and next steps (cf. Sect. 5).¹ A central theme emerged across all three phases: AI as a double-edge sword for TD; it is both a TD generator and a data-driven TD management tool. Discussions ranged from the quantification of AI-generated TD and tuning AI methods and models to avoid the introduction of new TD, to using AI to suggest improvements to manage TD, finding best practices to use AI to manage diverse TD (e.g., architectural, security) and reflecting upon the socio-technical implications of AI. As concrete priorities to be addressed in the years to come, participants

¹ The information of the workshop including its program is available at <https://conf.researchr.org/track/xp-2026/xp-2026-international-workshop-on-ai-for-technical-debt-management-in-agile-software-development--ai4td>.

agreed that: AI-generated TD should be treated as a first-class concern; the classical notion of TD should be extended with cognitive, intent, and harness debt; and AI use should be embedded within disciplined XP practices and PDCA-style governance (cf. Sect. 6).

2 Insights from the Accepted Papers

AI4TD followed a single-blind review process. Each submission received at least three reviews from the members of the program committee. Following the review submission, there was an online discussion period. The organizing committee together made the final decision on acceptance. Out of four submissions, two papers were accepted and presented at the workshop. Yngve Lindsjörn presented the paper entitled *Reducing Technical Debt with Generative AI: A Case Study of AI-Assisted Refactoring* by Iselin Mordal Bakke, Viktoria Stray, and Yngve Lindsjörn. The talk reported on a case study with 46 agile student teams: 23 used GPT and 23 used Gemini in Android Studio to iteratively refactor two Kotlin files each (19,404 LOC total). Using SonarQube metrics (code smells, technical debt, complexity, size) and a post-activity survey on usefulness and trust, the study found that GenAI-assisted refactoring substantially reduced code smells, technical debt estimates, and cognitive complexity, while slightly increasing cyclomatic complexity. Because this work focused mainly on the quantitative results, the discussion was around what qualitative examples of refactoring outputs could be to strengthen the evaluation and make the findings more interpretable. The presenter acknowledged that this point is important for future work. The presenter also clarified what large language models (LLM) versions used in this study.

The second paper is *From Smell to Solution: Multi-Agent LLMs Analyzing Cyclic Dependencies for LLM-enabled Refactoring* by Henrik Brunvatne Olafsen, Adela Nedisan Videsjorden, Arda Goknil, and Sagar Sen. The talk introduced a multi-agent LLM framework that turns static dependency data into verifiable explanations of cyclic architectural dependencies. The framework builds a dependency graph, detects strongly connected components, extracts minimal cycles as evidence, and uses hierarchical LLM agents to explain each cycle's location, structure, and impact, including where it can be safely resolved. Evaluation on seven open-source Python projects shows that these evidence-linked analyses effectively guide LLM-based refactoring, shrinking cyclic regions and improving structural quality without breaking builds or tests. The audience also got the clarification on what LLM versions used in the study (Qwen3-Coder-30B-A3BInstruct, a mid tier model, optimized for software engineering tasks). It is important to understand better how powerful LLMs needed to get the results that are presented in the paper. Another question from the audience was about the correctness of the refactoring recommendations that are very specific to Python stack. The Python-specific recommendations were evaluated by the authors as moderate measurable improvements without introducing new issues. For example, they shared that the Ruff linting issues were slightly increased mainly because of the stylistic convention differences with the tool's default rule

set rather than a true reduction in code quality. Meanwhile, the maintainability index was significantly increased by +0.70 with analysis (vs. +0.31 without).

3 Keynotes Takeaway

To set the theme for the workshop and initiate discussions, AI4TD featured two keynote talks on a broader view on AI and TD.

Sustainable Development and Technical Debt in the Age of GenAI: A Primer for XP Practitioners: Joseph (Joe) Yoder is a software architect, pattern author, and mentor, founder of The Refactory, and research collaborator at IME/USP. He coauthored Cloud Application Architecture Patterns and A Scrum Book, is known for the “Big Ball of Mud” pattern, and is an ACM Distinguished Member and Hillside Group Fellow. Graziela Simone Tonin is Associate Dean of Computer Engineering with over 20 years in technology, innovation, and startups in Brazil and the United States. She holds a PhD in Computer Science from IME-USP, has received an IBM research award, chaired national agile and software engineering workshops, and regularly speaks and consults in Brazil and abroad.

Presentation: The presentation focused on how Generative AI (GenAI) is changing the way teams manage technical debt, especially in Extreme Programming (XP) environments that emphasize rapid delivery and continuous change. It began with a brief review of what technical debt is and why it accumulates, then explored how GenAI can help teams identify, analyze, and address technical debt more efficiently and at greater scale. At the same time, the talk emphasized that GenAI introduces new risks, such as subtle defects, architectural drift, and loss of shared understanding, that can themselves become additional forms of technical debt if not actively managed. Framed by XP principles like feedback, simplicity, and collective ownership, the presentation described how GenAI can both improve visibility into technical debt and support better prioritization and faster repayment, provided it is used with disciplined oversight. It concluded with practical guidance for teams on sustaining development in an AI-augmented environment while avoiding AI-induced Technical Debt.

A Plan-Do-Check-Act (PDCA) Framework for Addressing AI-Generated Technical Debt: Ken Judy is a Senior Partner at Stride in New York, where he modernizes legacy systems with AI and advanced compiler techniques. He focuses on turning AI adoption into measurable business outcomes through scalable governance, daily AI-assisted development, and XP-informed Plan-Do-Check-Act practices for AI code generation.

Presentation: The presentation examined how AI-assisted code generation is contributing to a growing sustainability problem in software development. Drawing on GitClear data, it highlighted a roughly tenfold increase in duplicated code blocks since 2022, and referenced DORA research that links AI adoption with persistent delivery instability across organizations. The speaker argued that the underlying issue is structural: when AI tools are adopted in an unstructured

way, they maximize short-term output at the expense of long-term maintainability and stability. To address this, the talk proposed applying core Extreme Programming (XP) disciplines within a Planâ€šDoâ€šCheckâ€šAct (PDCA) framework at the level of humanâ€šAI collaboration. This structured approach aims to close the accountability gap left by unstructured AI coding practices. The presentation outlined this framework, reviewed supporting empirical evidence, and described a tutorial component in which participants practiced specific intervention behaviors while working on AI-assisted coding tasks.

4 Industrial Talks

The workshop featured two talks from industry. First, Mili Orucevic and Maren Maritsdatter Kruke from Visma software international AS gave a talk on the *Challenges in Industry working with AI and Technical Debt*. TD used to be a discipline problem (teams knew what was wrong but couldn't find time to fix it), but AI has transformed it into a comprehension problem (teams don't even realize debt is accumulating because code might arrive too fast). The presentation contrasts the pre-AI and post-AI worlds across several dimensions. Before AI, debt accumulated gradually, developers understood the code they wrote, ownership was clear, and code review kept pace with output. After AI, accumulation is explosive, developers accept code they only partially understand, ownership can be blurred between human and machine, and reviewers face cognitive overload from high-volume AI-generated PRs. External data backs this up—AI PRs show 1.4x more critical and 1.7x more major issues than human-written code in one of the studies. In the industry, on the increase side (AI increases TD), practitioners note that AI lacks architectural context, junior developers push more code without deep understanding, and legacy codebases don't play well with LLMs. On the decrease side (AI decreases TD), senior developers are successfully using AI for large-scale migrations, code reviews, test coverage, and modernization efforts. A .NET migration completed with an 80% reduction in planned time, 30-year-old on-prem product rebuilt as SaaS with 68% time reduction, and a team of 5 developers now generating the output of 16 using multi-agent development. The takeaway: AI is simultaneously changing how we perceive TD, accelerating its accumulation, and clearly helping teams remove more of it than ever before. The impact—good and bad—is already real, and the key variable is the skill level and judgment of the humans using the tools.

Looking more to the future, Jan Henrik Gundelsby from Knowit shared his thoughts in the *future research about AI and TD*. Building on Storey's recent work [2], this talk proposed a Debt Model (Fig. 1.) that extends TD to capture how AI-assisted development reshapes what accumulates, where, and at what cost. TD, as Cunningham defined it back in 1993 [1], lives in code and accumulates when implementation decisions compromise future changeability. It is well-studied, with mature tools and metrics. Cognitive debt lives in people and accumulates when shared understanding erodes faster than teams can replenish it [2]; when AI generates code faster than teams can absorb it, speed without

understanding becomes unsustainable. Intent debt lives in externalized knowledge; the rationale that developers and AI agents alike need, but that no one has captured [2]. The talk extended this framing with a fourth concept: *harness debt*; the debt that accumulates when organizations lack the structures (tools, guardrails, monitoring, workflows, governance) required to control AI-generated code and autonomous agents. Storey [2] already places tooling and AI agents inside cognitive debt, treating them as part of where shared understanding lives and must be repaid.

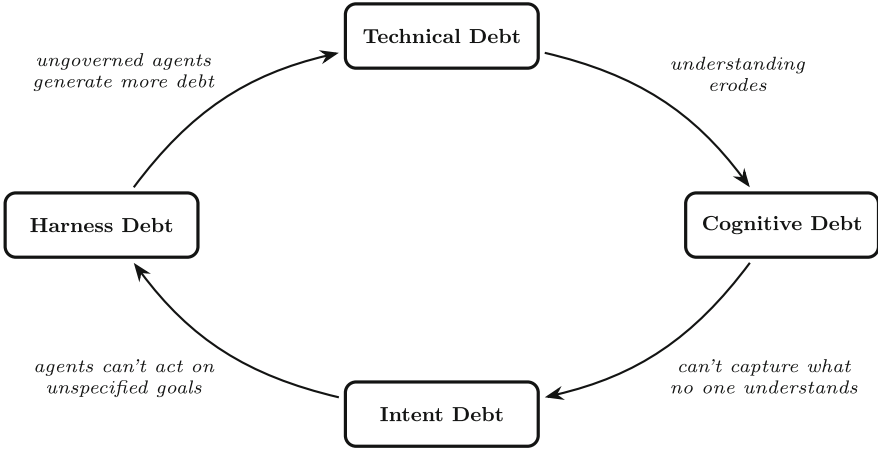


Fig. 1. The debt cycle: Four interacting debt types in AI-era software systems.

Elevating harness debt to a first-class category recognizes that the governance infrastructure has its own dynamics: ungoverned agents generate debt regardless of whether the team’s understanding is intact, and misaligned guardrails compound faster than human review can catch. Where cognitive and intent debt concern understanding and rationale, harness debt concerns the governance mechanisms themselves. Together, the four form a cycle: ungoverned agents generate more TD, TD erodes understanding, eroded understanding cannot externalize intent, and missing intent leaves agents unable to act on unspecified goals. Furthermore, the presentation outlined seven directions for future research: measuring invisible debt (metrics for cognitive, intent, and harness debt); evolving teams with AI (skill degradation, learning paths, roles, decision-making); socio-technical practices (shared understanding, pairing, reviews, context engineering); AI-generated code debt (smells, vulnerabilities, over-engineering, duplication); AI system and governance debt (LLM evaluations, model updates, shadow AI use, harness debt as governance infrastructure); ecosystem and monoculture (what happens when development depends on shared AI models); and AI-augmented debt management (AI-assisted prioritization and remediation). The closing argument positioned these as both a research agenda and a call for tighter

industryâ–academia collaboration, given how quickly AI-augmented practice is outpacing the methods used to study it.

5 Group Discussion and Fishbowl Panel

AI4TD is an open workshop, i.e., participation is not limited to presenters but were open to all who are interested in the workshop topic. We organized group discussions about challenges and future research. The topics were chosen as they emerged during the workshop, and were discussed for 45 minutes in an open space by the participants.

Group Sessions and Topic Discussed:

Incrementing or Reducing TD with AI? This question asks whether using AI in software development is increasing technical debt (TD) or helping to pay it down. It matters because AI can accelerate coding but also introduce hidden design flaws, duplication, and complexity. Understanding which effect dominates is critical for deciding how to adopt AI safely, sustainably, and with the right engineering and governance practices in place. The main discussion was around the need to assess new Functional and Non-Functional Requirements for AI-generated code.

Cognitive-, Intent-, Comprehension Debt elaborated on how much extra mental effort teams must spend to understand code (cognitive), what it was meant to do (intent), and how it actually works end-to-end (comprehension). It's important because even if systems "work," growing debt in these areas slows onboarding, increases errors, and makes change risky and expensive over time. The discussion revolved around alignment between human and AI mental models, the trust of both AI system and people using such systems and different costs of various scenarios, for example building fast but losing comprehension etc. What is the right tradeoff?

Change in Perception of TD looked at how AI adoption is reshaping what teams consider to be TD and how serious they think it is. As AI makes it easy to produce large amounts of code quickly, some teams may underestimate long-term design and maintenance costs, while others start to see new AI-specific debts (e.g., prompt, model, or data debt). This shift in perception is crucial because it drives how organizations govern AI use, prioritize refactoring, and balance speed against long-term sustainability.

Socio-Technical Practices to Manage TD in AI focused on how not just tools and code, but also people, processes, and team structures are used to control TD created or affected by AI. It's important because AI-driven code and systems can only stay sustainable if we pair good engineering practices (tests, architecture, automation) with social ones (pairing with AI, code reviews, shared standards, governance) that keep teams aligned, accountable, and able to safely evolve AI-augmented systems.

Harnessing Agents, Standardizing vs Innovating was about how teams use AI and agentic tools in a way that balances consistency with experimentation. It

was considered important because standardizing how agents are used (patterns, guardrails, workflows) can reduce TD and risk, while leaving room for innovation allows teams to discover new, higher-value uses of AI—without creating chaos or unsustainable systems.

How Do We Change the Way We Learn Technical Debt and AI? asked how education and training must evolve so people not only learn to use AI tools, but also learn to recognize, measure, and manage the TD they create or reshape. Current learning focuses on speed and features, while sustainable AI-augmented development also depends on a stronger emphasis on requirements, clear goals, constraints, and value, and on improving students' ability to think at higher levels of abstraction and maintain an overview of systems besides assessing AI-generated code. Without these shifts, teams risk fast AI-driven development that is misaligned, fragile, and hard to evolve.

Legacy, Modernization, TD and AI looked at how AI changes the way we deal with legacy systems and their TD during modernization. AI can accelerate migration, refactoring, and remediation, but can also layer new debt on top of old if used naively. Understanding this relationship helps teams decide when and how to apply AI so modernization actually reduces long-term risk and cost instead of just reshaping or hiding existing legacy problems.

The Fishbowl Panel was initiated by Mili Orucevic Visma, Maren Maritsdatter Kruke Visma software international AS, Jan Henrik Gundelsby Knowit, Joseph (Joe) Yoder The Refactory, Inc., Graziela Simone Tonin INSUPER, and then joined in by other participants in the room.

The panel explored how sociotechnical practices in software development are changing with AI. On one side, AI is making agile development more powerful—for example, enabling challenging legacy migrations (such as from PHP to C#) that are now more feasible with the right prompts and skills. On the other side, it is becoming harder to track and monitor what really matters, especially costs and making TD more visible and traceable.

From an industry perspective, participants stressed the need for strong practices to mitigate TD in the age of AI and highlighted that deliberately harnessing TD can be valuable. The panel agreed that how humans collaborate around software development is critical, and that new standards, playbooks, and practices will be needed. They also noted the risk of using AI as a convenient scapegoat and emphasized the importance of education and human-in-the-loop approaches.

Questions from the developers point of view, such as whether they still need to learn or whether AI might replace their jobs, were seen as central to this transition. The discussion highlighted the need to slow down to understand before accelerating adoption, the role of upskilling at scale, the importance of bounded contexts for effective AI use, and the value of actively cultivating a strong AI adoption culture.

6 Conclusion

In summary, the workshop showed that AI is fundamentally reshaping TD, not only in code but also in understanding, intent, and governance. GenAI and AI-assisted development can dramatically accelerate modernization and debt repayment, yet they also introduce new, hard-to-see forms of debt and amplify existing ones when adopted without structure. Across keynotes, industrial talks, group sessions, and the fishbowl panel, participants converged on several priorities: treating AI-generated TD as a first-class concern; extending classical TD concepts with cognitive, intent, and harness debt; and embedding AI use within disciplined XP practices and PDCA-style governance. The community emphasized socio-technical responses, for example new standards, playbooks, education, and large-scale upskilling, to keep humans in the loop, preserve shared understanding, and align rapid AI-driven change with sustainable value. These discussions define a research agenda that spans measurement, practices, education, and legacy modernization, and call for tight collaboration between industry and academia to keep pace with AI-augmented software development.

Acknowledgments. The organizers of this workshop from Norway have received funding from the Research Council of Norway's Innovation Project for the Industrial Sector (IPN) Programme under grant agreement for project No. 340991 (TechDebtOps).

References

1. Cunningham, W.: The WyCash portfolio management system. ACM SIGPLAN OOPS Messenger 4(2), 29–30 (1993). <https://doi.org/10.1145/157710.157715>
2. Storey, M.-A.: From technical debt to cognitive and intent debt: rethinking software health in the age of AI. arXiv preprint [arXiv:2603.22106](https://arxiv.org/abs/2603.22106) (2026). <https://doi.org/10.48550/arXiv.2603.22106>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





From Smell to Solution: Multi-agent LLMs Analyzing Cyclic Dependencies for LLM-Enabled Refactoring

Henrik Brunvatne Olafsen¹ , Adela Nedisan Videsjorden² ,
Arda Goknil² , and Sagar Sen²

¹ University of Oslo, 0371 Oslo, Norway
henrbol@uio.no

² SINTEF Digital, 0373 Oslo, Norway
{adela.videsjorden,arda.goknil,sagar.sen}@sintef.no

Abstract. Cyclic dependencies are a persistent architectural smell that undermines modularity, blurs ownership, and elevates maintenance risk. While Large Language Models (LLMs) provide refactoring abilities, existing tools struggle to address smells without intervention evidence. We introduce a multi-agent LLM framework for cyclic dependency analysis, which transforms static dependency evidence into structured, verifiable explanations of architectural entanglement. The approach constructs a structural dependency graph, detects cyclic regions via strongly connected components (SCC), extracts cycles as minimal witnesses of cyclicity, then produces grounded analyses. These are produced by a hierarchical team of LLM agents that describe the position, structure, and architectural impact of each cycle, clarifying why it exists and where it can be safely resolved. Evaluation on seven open-source Python projects shows that evidence-linked analyses can guide LLM-driven edits, reducing strongly connected components size and improving structural quality.

Keywords: Architectural Debt · Software Engineering · Generative AI

1 Introduction

Cyclic dependencies often arise from hasted design choices leading to regions of code becoming circularly dependent [12]. Their presence erodes modularity: builds and tests grow fragile, small edits cause cascading changes, and architectural boundaries blur [6]. Existing detectors^{1,2} report that a cycle exists but rarely indicate *where* to intervene or *why* a specific cut is meaningful.

LLM-based refactoring tools can automate the refactoring process but falter without grounded, task-specific context [3, 9]. Particular tools list mutually dependent modules but rarely indicate where the cycle originates or how to

¹ Arcan: <https://www.arcantech/> (Accessed 2025-10-08).

² Understand: <https://scitools.com/> (Accessed 2025-10-08).

remove without side effects [11]. Guiding refactoring tools with grounded analyses that identify the structural cause of cyclicity can enable automated systems to safely intervene and validate improvements through measurable architectural feedback.

Our approach involves the generation of actionable analysis or explanations that guide autonomous refactoring. We define “explanation” operationally as a structured reasoning trace linking structural evidence of cyclicity to its cause. The process begins with constructing a structural dependency graph, then modeling cyclicity via SCCs, namely components where architectural entanglement is most severe. We then extract minimal representative cycles as compact witnesses of entanglement. A dynamically-scalable team of LLM-based agents generates the structured analysis of the cyclic dependency. The analysis and code are submitted to a refactoring framework for sandboxed and test-driven code edits.

We evaluate the approach guided by two Research Questions (RQs), assessing the *effectiveness* and *impact* of analysis-guided refactoring:

- **RQ1:** *Do evidence-linked analysis of cyclic dependencies increase the effectiveness of autonomous code refactoring at eliminating these dependencies?* By answering RQ1, we capture the effect of using the generated analysis on refactoring success by comparing outcomes *with* and *without* analysis.
- **RQ2:** *How does analysis-guided refactoring change overall code quality, as reflected by standard maintainability and design indicators?* By answering RQ2, we assess the impact of the analysis-driven refactoring on code quality, zooming on standard quality indicators compared *with* and *without* analysis.

Our paper contributes with a pipeline that transforms structural dependency evidence into verifiable architectural improvements by integrating static analysis, multi-agent LLM reasoning, and sandboxed execution; a hierarchical multi-agent LLM system that produces grounded analyses of cyclic dependencies; and the evaluation showing that evidence-linked analyses improve refactoring success (by $\approx 62\%$), reduce coupling and SCC size, and enhance maintainability.

The paper is organized as follows: Section 2 outlines the approach. Section 3 presents the evaluation and results, while Sect. 4 discusses results. Section 5 reviews related work, and Sect. 6 concludes the paper.

2 Overview of the Approach

Our four-stage pipeline (Fig. 1) transforms static dependency evidence into verified architectural improvements through structural analysis, cycle detection, cycle analysis, and autonomous refactoring. Stages 1–4 follow in greater detail.

Static Dependency Graph Construction (Stage 1). Stage 1 extracts structural dependencies of the given repository as a directed graph, forming the basis for cycle detection. Repositories are parsed using static a dependency analysis

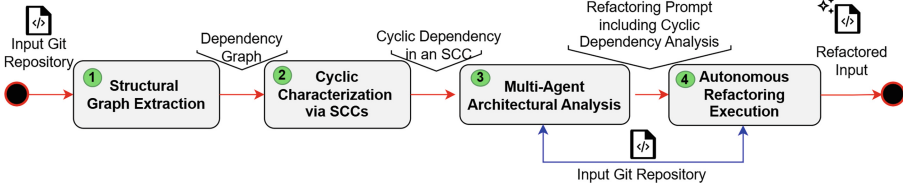


Fig. 1. Overview of the four-stage pipeline.

tool (pydeps³) into graphs where each node denotes a class, module, or package, and each directed edge a static dependency. The graph captures import-level relations such as *Import*, *Implement*, *Include*, *Extend* and *Mixin*, retaining *structural dependencies*, excluding runtime or usage links.

Cyclicity Characterization via SCCs (Stage 2). Stage 2 identifies cyclic dependencies in the dependency graph produced by Stage 1. Rather than enumerating all simple cycles (to avoid overlaps), we analyze SCCs, each a maximal subgraph where every node is reachable from every other.

From Structural Graph to SCCs. Let the directed *structural* dependency graph be $G = (V, E)$, where E contains only edges extracted in Stage 1, and V is the set of vertices representing classes/modules/packages. We partition V into SCCs $\{S_1, \dots, S_k\}$ using Tarjan’s algorithm [16] in $O(|V| + |E|)$ time. For an SCC $S \subseteq V$ define the set of *internal* edges:

$$E_S = \{(u, v) \in E \mid u \in S, v \in S\},$$

and record also the number of *inter*-SCC edges crossing S (incoming/outgoing). These quantities provide transparent indicators of architectural entanglement.

SCC Metrics and Selection Policy. We characterize each SCC S by three simple indicators: Size $|S|$, Amount of Internal edges $|E_S|$, and Inter-SCC coupling (Ratio of in/out crossings). Because E already filters to *structural* dependencies, all internal edges are treated *equally*. To prioritize analysis, we adopt:

$$S^* = \arg \max_S (|S|, |E_S|), \quad (1)$$

i.e., select the *largest* SCC; if tied on node count, select the one with more internal edges, as larger tangles often signal higher coupling and risk [8].

Representative Cycle Extraction. Within S^* we extract a *representative cycle*, i.e., a minimal witness of cyclicity. Let $\mathcal{C}(S^*)$ be the set of simple cycles and let

³ <https://github.com/thebjorn/pydeps> (Accessed 2025-10-08).

$L(C)$ be the cycle length. We first choose a shortest cycle:

$$\mathcal{C}_{\min} = \{ C \in \mathcal{C}(S^*) \mid L(C) = \min_{C'} L(C') \} \quad (2)$$

$$C_r = \arg \max_{C \in \mathcal{C}_{\min}} \sum_{v \in C} \text{cent}_{S^*}(v) \quad (3)$$

We then select the representative cycle C_r from $\mathcal{C}_{\min}$ (Eq. 2) by favoring higher aggregate node centrality computed *within* the induced subgraph of S^* (Eq. 3), where cent_{S^*} can be degree or betweenness centrality restricted to S^* . Prioritizing high-centrality paths targets influential connectors, removing any edge on C_r often dissolves or shrinks S^* with a small edit (desirable for Stage 4).

Table 1. Overview of the open-source Python repositories in the study.

ID	LoC (K)	Modules	Deps.	SCCs	% Mods in SCCs	Max SCC	Sel.
P1	18.2	79	374	1	7.6%	6	2
P2	8.9	17	63	3	52.9%	5	2
P3	19.4	57	240	2	14.0%	6	2
P4	23.5	78	424	1	46.2%	36	8
P5	11.7	25	99	1	68.0%	17	4
P6	35.4	164	864	1	1.2%	2	1
P7	8.0	36	162	1	5.6%	2	1

Multi-agent Architectural Analysis (Stage 3). Stage 3 converts the structural findings from Stage 2 into an evidence-linked analysis using LLM-driven agents. The input consists of the identity of S^* (its nodes, $|S^*|$, $|E_{S^*}|$), the C_r (ordered nodes/edges), and optional in-SCC centrality scores used for the tie-break in (3). Each file in the representative cycle is analyzed by paired **Source-side and Target-side Analyst** agents. Each source-side analyst extracts concrete evidence of coupling, while each target-side analyst classifies the imported entities by their architectural importance. The **Cycle Analyst** agent aggregates these findings into a coherent, evidence-grounded explanation describing how the cycle forms, which edges carry the strongest architectural weight, and how they influence modular boundaries. The **Verifier** agent cross-checks all extracted dependencies against the structural graph from Stage 2, ensuring that the explanation remains factually consistent and traceable to observable code-level evidence. The system dynamically scales the number of agents based on the cycle size.

Autonomous Refactoring Execution (Stage 4). Stage 4 converts the analysis from Stage 3 into concrete edits. Using OpenHands [17] as the backend, the LLM agent edits code, runs builds and tests, and logs all actions. After execution, Stages 1–2 are rerun to assess architectural impact.

3 Evaluation

Subjects of the Study. We evaluated 7 open-source Python projects from diverse domains. Projects were selected based on (i) active maintenance within the past year, (ii) presence of a functioning test suite, (iii) non-trivial size (>5 KLoC), and (iv) a clear module structure. Table 1 summarizes key characteristics: “SCCs” is the number of SCC (size ≥ 2), “% Mods in SCCs” is the proportion of modules involved in cyclic dependencies, “Max SCC” is the size of the largest cycle and “Sel.” is the number of selected cycles.

Experimental Setup. A total of 20 cycles were selected, their distribution reflecting a balanced mix of small and large cyclic structures. Each cycle was processed eight times, and results were averaged to reduce stochastic variation. We used *Qwen3-Coder-30B-A3B-Instruct* with a temperature of 0.1, balancing reasoning performance and stability for code transformation tasks. Experiments were executed on a GPU cluster (NVIDIA A100/H100), with each run allocated one GPU, four CPU cores, and 48 GB RAM via SLURM.

Configurations and tool versions are included in the **replication package**⁴.

Table 2. Success and architectural improvement (with vs. without analysis).

Condition	Success%	$\overline{\Delta}$ Edges	$\overline{\Delta}$ Nodes	$\overline{\Delta}$ LOC	p_{succ} (GLMM)	Odds ratio
With	29.38	-5.40	-1.66	-792.6	0.009*	2.15
Without	18.13	-2.17	-0.90	-316.0	—	—

Table 3. Breakdown of refactoring outcomes (with vs. without analysis)

Condition	Success	Behavior regressed	Structure not improved	Both failed	Other error
With	29.4%	20.0%	39.4%	9.4%	1.9%
Without	18.1%	16.2%	45.6%	17.5%	2.5%

3.1 Results

RQ1: Effectiveness of Evidence-Linked Analysis for Autonomous Refactoring of Cyclic Dependencies. We assess the effectiveness of analysis-guided refactoring by measuring both refactoring success and structural improvement. A refactoring is a *success* if it both preserves behavior (no new test failures

⁴ <https://figshare.com/s/25633b5e677d1219df39>.

Table 4. Per-project architectural improvements (with vs. without analysis).

Metric	Cond.	P1	P2	P3	P4	P5	P6	P7
Success%	With	25.0	50.0	6.3	18.8	59.4	12.5	25.0
	Wout	12.5	6.3	12.5	17.2	21.9	25.0	50.0
Δ Edges	With	-6.25	-2.50	-3.00	-4.33	-7.79	-2.00	-2.00
	Wout	-4.50	-2.00	-2.50	-2.55	-1.00	-2.00	-2.00
Δ Nodes	With	-3.75	-1.25	-1.00	-0.75	-1.95	-2.00	-2.00
	Wout	-2.50	-1.00	-0.50	-0.64	0.00	-2.00	-2.00
Δ LOC	With	-1397	-397	-652	-105	-1235	-1588	-763
	Wout	-719	-419	-326	-39	1	-1588	-763

in *pytest*) and achieves a global structural improvement, defined as a reduction in the total number of inter-module dependency edges (Δ Edges < 0) when summed across all SCCs. Structural improvement is further quantified using differences in the number of inter-module edges (Δ Edges), nodes (Δ Nodes), and lines of code (Δ LOC), computed as post-pre values over all SCCs. Statistical significance is assessed using mixed-effects logistic regression, or binomial generalized linear mixed models (GLMMs) [2] with a logit link because the outcome is binary and observations are clustered by repository and dependency cycle.

To address *RQ1*, we compare success rate and structural improvement with and without analysis. Analysis-assisted refactorings achieved a 29.4% success rate, surpassing 18.13% without analysis (Table 2). Successful runs also yielded stronger structural improvements—on average Δ Edges = -5.40 vs. -2.17, Δ Nodes = -1.66 vs. -0.90, and Δ LOC = -792.6 vs. -316.0—producing leaner, more modular graphs. The outcome breakdown in Table 3 clarifies this effect. Each run is classified by whether it preserved test behavior and/or improved global structure. “Behavior regressed” indicates new *pytest* failures despite structural improvement, while “Structure not improved” indicates no edge reduction, yet tests have passed. “Other error” covers runs that did not complete. The share of runs that failed both behaviorally and structurally decreased (17.5% to 9.4%) with analysis guidance. We observe lower values for failed cases due to structure not being improved, or where other errors took place, where the analysis guidance were provided. The reduction in “Structure not improved” outcomes (45.6% to 39.4%) shows that analysis guidance reduces wasted actions on non-critical dependencies. The proportion of failed runs where behavior regression is observed (20.0% with and 16.2% without) shows that while structural improvement is made, analysis guidance may lead to test failures.

Table 4 shows that the strongest effects were recorded in medium repositories (P2, P5) where multiple refactoring alternatives exist. These two projects had the least amount of modules and recorded dependencies. Highly entangled projects (P3, P7) showed better refactoring success without rationale-guided analysis, indicating that project-specific factors may cause result variations.

Table 5. Change in code quality for successful refactorings (with vs. without analysis).

Cond.	Δ Ruff ↓	Δ MI ↑	Δ CC ↓	Δ PyExam A ↓	Δ PyExam C ↓	Δ PyExam S ↓	Δ Bandit ↓
With	+1.17	+0.75	0.00	-4.60	0.00	-0.17	0.00
Without	+1.90	+0.35	0.00	-5.83	0.00	0.00	0.00

RQ1 Conclusion: Analysis-guided refactoring outperformed the baseline, yielding higher success and stronger structural improvements in most runs. Averaged across projects, it removed a larger number of cyclic dependencies and produced leaner, more modular architectures while preserving behavior.

RQ2: Impact of Autonomous Cycle-Breaking Refactoring on Overall Code Quality. We evaluate the impact of successful, behavior-preserving refactorings on overall code quality relative to their pre-refactoring baselines. We compute post-pre differences (Δ) across multiple automated indicators: linting issues from *Ruff*⁵ (fewer is better), *Maintainability Index* (MI) and *cyclomatic complexity* from *Radon*⁶ (higher MI, lower *D-F* complexity) and security warnings from *Bandit*⁷ (fewer high-severity alerts). We include quality metrics from *PyExamine*⁸, grouped into *Architectural* (A), *Code-level* (C), and *Structural* (S) categories. RQ2 uses linear mixed-effects models (LMMs) [4] for continuous post-pre quality-metric deltas.

To respond to RQ2, we examined refactoring impact on overall code quality and maintainability compared to baselines. Table 5 summarizes mean post-pre deltas (Δ) per successful refactoring. Improvement translates to achieving negative values for Ruff, CC, PyExam, Bandit and positive values for MI. Table 6 reports mixed-effects tests for change from baseline. Refactoring improved maintainability and code hygiene, with analysis-assisted runs yielding slightly stronger effects. The MI rose (+0.75 vs +0.35). This indicates that analysis-guided refactoring improves static analyzers’ maintainability assessment. Ruff linting issues increased for both conditions (Δ Ruff = +1.17 with and +1.90 without analysis), however less for guided refactoring. Ruff increase should be interpreted cautiously, as it likely reflects stylistic convention differences rather than reduced code quality. Other metrics, such as cyclomatic complexity and static security warnings (Bandit) remained stable, preserving behavior.

Analyses using PyExamine revealed consistent reductions in architectural-level issues ($\Delta \approx -4.6$ with the analysis guidance vs $\Delta \approx -5.83$ without). We

⁵ <https://github.com/astral-sh/ruff> (Accessed 2025-10-08).

⁶ <https://pypi.org/project/radon> (Accessed 2025-10-08).

⁷ <https://github.com/pycqa/bandit> (Accessed 2025-10-08).

⁸ https://github.com/KarthikShivasankar/python_smells_detector (Accessed 2025-10-08).

highlight that the analysis-assisted refactoring achieves slightly better results (with $\Delta \approx -0.17$ for structural metrics), which is consistent with refactoring focused on inter-module edges.

RQ2 Conclusion: Refactorings improved maintainability while preserving correctness and security. Explanation-guided runs achieved higher maintainability gains (+0.75 vs. +0.35) but showed a modest increase in linting issues. Other metrics remained stable, confirming behavior preservation.

Threats to Validity. Our evaluation is limited to Python and may not generalize to other languages. Results can be affected by LLM non-determinism; to mitigate this, experiments were repeated 8 times with fixed seeds. Statistical validity may be constrained by sample size and SCC interdependence, therefore, we used paired, nonparametric tests with effect. Selected quality metrics may only provide partial coverage and alternative LLMs and refactoring tools may yield different results, aspects we acknowledge and plan to address in future work.

4 Discussion

Evaluation under **RQ1** shows that providing analysis guidance improves the effectiveness and reliability of LLM-driven refactoring, even when using a mid-sized LLM. Across seven open-source repositories and twenty representative cycles, explanation-assisted runs achieved higher success (29.4%) than the baseline (18.1%), with statistically significant gains in structural improvement. While success rates remain modest, this outcome was expected given our deliberate use of a smaller model with limited model capacity. The observed improvement thus reflects the explanatory mechanism’s contribution rather than raw model strength. As shown for **RQ2**, successful refactoring preserved test behavior and produced consistent maintainability gains without increasing complexity or risks. Although Ruff linting counts rose slightly, this likely reflects stylistic mismatches with Ruff’s defaults rather than poorer code quality. Overall, the approach yields safe, quality-neutral to positive edits.

Table 6. Mixed-effects tests for change per metric (with vs. without analysis).

Metric	p vs 0 (Without, $n=29$)	p vs 0 (With, $n=47$)
Ruff issues	0.0029	0.0030
MI avg	0.0497	0.0027
CC (D+E+F)	1.000	1.000
PyExam (A/C/S)	0.373/1.000/0.736	0.567/1.000/0.511
Bandit (high)	1.000	1.000
mypy errors	—	—

Implications. Although performance remains constrained by model capability, the results indicate that explicit analysis guidance can substantially close the gap between naïve automation and explainable, architecture-aware refactoring. While the evaluated projects are relatively moderate in LoC size, the approach can scale to larger systems by dynamically generating source and target analyzers. This contrasts with relying on a single LLM, where small context windows can quickly become a limiting factor. The approach is suited for integration with continuous integration/continuous deployment pipelines and code review processes by generating separate pre-merge refactoring patches.

5 Related Work

Early research on behavior-preserving refactoring introduced algorithmic planning beyond ad-hoc edits [10]. Search-based methods later framed refactoring as a multi-objective optimization problem often integrating developer feedback and explainable operators for guided, auditable recommendations [1]. Tool-supported frameworks such as ARIES further automate higher-level transformations like Extract Class or Extract Package, improving cohesion without increasing coupling [13]. Recent work integrates LLMs with IDE and tool safeguards to perform verified, behavior-preserving refactorings. EM-Assist [14, 15] automates Extract Method in IntelliJ by combining LLM-generated candidates with static validation and IDE-based execution, showing that precise intents and safeguards enable reliable automation. UTRefactor [5] applies this to test code, using contextual reasoning and checkpointed workflows to remove test smells across OSS projects. Repository-level benchmarks like SWE-bench [7] and systems such as SWE-agent [18] extend this to full repositories, coupling LLMs with execution environments for code editing and testing.

In contrast, our work targets cyclic dependencies at the architectural level, linking architectural analysis to automated execution by analyzing structural dependency graphs, characterizing entanglement via SCCs, generating evidence-linked analyses, and closing the loop through executable, measurable edits.

6 Conclusion

This paper introduced an analyze-to-refactor approach for cyclic dependencies, combining structural dependency analysis, multi-agent LLM reasoning, and autonomous refactoring. The evaluation shows that analysis-guided refactoring consistently outperforms baseline approaches in effectiveness, yielding higher success rates and stronger structural improvements in most runs. We show that analysis-informed refactoring improves maintainability without compromising correctness or security, with only minor increases in linting issues.

Future work will explore a larger variety of LLMs, other programming languages (such as C#) and projects with greater complexity and broader scope. Other research avenues include hybrid validation where developers can provide feedback on the refactoring suggestions, investigating refinement loops beyond one-shot resolutions and extending the approach to other architectural smells.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alizadeh, V., et al.: An interactive and dynamic search-based approach to software refactoring recommendations. *IEEE Trans. Software Eng.* **46**(9), 932–961 (2018)
2. Clayton, D.G.: Generalized linear mixed models. *Markov chain Monte Carlo Pract.* **1**, 275–302 (1996)
3. Cordeiro, J., Noei, S., Zou, Y.: LLM-driven code refactoring: Opportunities and limitations. In: 2025 IEEE/ACM Second IDE Workshop (IDE), pp. 32–36. IEEE (2025)
4. Fox, J.: Linear mixed models. Appendix R S-plus Companion Appl. Regression **16**, 2349–2380 (2002)
5. Gao, Y., Hu, X., Yang, X., Xia, X.: Automated unit test refactoring. *Proc. ACM Softw. Eng.* **2**(FSE), 713–733 (2025)
6. Gnoyke, P., Schulze, S., Krüger, J.: Evolution patterns of software-architecture smells: an empirical study of intra-and inter-version smells. *J. Syst. Softw.* **217**, 112170 (2024)
7. Jimenez, C.E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., Narasimhan, K.: Swe-bench: can language models resolve real-world GitHub issues? *arXiv preprint [arXiv:2310.06770](https://arxiv.org/abs/2310.06770)* (2023)
8. Li, Z., et al.: Characteristics of multiple-component defects and architectural hotspots: a large system case study. *Empir. Softw. Eng.* **16**(5), 667–702 (2011)
9. Liu, B., et al.: An empirical study on the potential of LLMs in automated software refactoring. *arXiv preprint [arXiv:2411.04444](https://arxiv.org/abs/2411.04444)* (2024)
10. Mens, T., Tourwé, T.: A survey of software refactoring. *IEEE Trans. Software Eng.* **30**(2), 126–139 (2004)
11. Mumtaz, H., Singh, P., Blincoe, K.: A systematic mapping study on architectural smells detection. *J. Syst. Softw.* **173**, 110885 (2021)
12. Oyetoyan, T.D., Cruzes, D.S., Conradi, R.: A study of cyclic dependencies on defect profile of software components. *J. Syst. Softw.* **86**(12), 3162–3182 (2013)
13. Palomba, F., et al.: Extract package refactoring in Aries. In: IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, pp. 669–672. IEEE (2015)
14. Pomian, D., Bellur, A., Dilhara, M., Kurbatova, Z., Bogomolov, E., Bryksin, T., Dig, D.: Next-generation refactoring: combining LLM insights and ide capabilities for extract method. In: 2024 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 275–287. IEEE (2024)
15. Pomian, D., et al.: EM-assist: safe automated extractmethod refactoring with LLMs. In: Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, pp. 582–586 (2024)

16. Tarjan, R.E.: Data structures and network algorithms. SIAM (1983)
17. Wang, X., et al.: Openhands: an open platform for ai software developers as generalist agents. arXiv preprint [arXiv:2407.16741](https://arxiv.org/abs/2407.16741) (2024)
18. Yang, J., et al.: SWE-agent: agent-computer interfaces enable automated software engineering. In: Advances in Neural Information Processing Systems, vol. 37, pp. 50528–50652 (2024)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Reducing Technical Debt with Generative AI: A Case Study of AI-Assisted Refactoring

Iselin Mordal Bakke^{ID}, Viktoria Stray^{(✉) ID}, and Yngve Lindsjörn^{ID}

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

Abstract. Agile teams need fast, actionable feedback to manage Technical Debt (TD). However, refactoring is often postponed due to time pressure. Generative AI (GenAI) promises to accelerate refactoring, but its effect on maintainability, TD at the source code level, and code correctness is still unclear. We conducted a case study in a team-based agile course, where 46 student teams refactored Kotlin code using either GPT (23 teams) or Gemini in Android Studio (23 teams). Each team selected two files and applied iterative GenAI-assisted refactoring. We analyzed a total of 19,404 lines of code. We assessed changes using SonarQube metrics (code smells, TD estimates, complexity, and size metrics). We also analyzed responses from a post-activity survey on perceived usefulness and trust. GenAI-assisted refactoring reduced total code smells, SonarQube-estimated TD, and cognitive complexity, while total cyclomatic complexity increased, although average cyclomatic complexity per function decreased markedly.

Keywords: Refactoring · Static Analysis · SonarQube · Agile Software Development · Software Engineering Education · Technical Debt

1 Introduction

Technical Debt (TD) remains a central challenge in agile software development, where short iterations, evolving requirements, and continuous delivery pressure often prioritize rapid progress over long-term maintainability [14]. While refactoring is a well-established practice for managing TD, it is often postponed because it is perceived as time-consuming and teams prioritize delivering new features [6, ch 2]. Over time, such postponement can lead to increased maintenance costs, reduced readability, and lower development productivity [2, 11], particularly in team-based settings where multiple developers interact with the same codebase [3].

A study by Besker et al. [2] showed that TD reduces developer productivity and software viability, with developers reporting that approximately 23% of their working time is wasted due to TD. The study further found that TD also

increased the need for additional activities such as testing, source code analysis, and refactoring. TD is also shown to be contagious, as existing TD often leads to the introduction of further TD, indicating the importance of prioritizing refactoring [2]. AI-driven code generation and refactoring tools promise a significant improvement similar to what integrated development environments (IDEs) did in the 1990s. Many organizations use them to speed up feature delivery, cut costs, support maintenance, close skill gaps, improve code quality, and reduce TD. The most notable development in AI for code so far has been Generative AI (GenAI) assistants and chatbots [7].

Prior studies have explored the use of LLMs for automated code refactoring and have reported promising improvements in code quality. However, the literature also shows that LLM-generated refactorings can introduce functional changes or fail existing tests [5, 13]. To further investigate this, we conducted a case study in a team-based agile educational setting, where student teams used GenAI tools to refactor existing code. Using SonarQube as a static code analyzer, we evaluate changes in code smells, complexity, size metrics, and estimated TD at the source code level before and after GenAI-assisted refactoring. In addition, we analyze students' perceptions of GenAI as a refactoring tool. By combining quantitative SonarQube measurements and metrics with developer perceptions, this study provides early empirical evidence on the role of GenAI in TD management within agile development in a student setting. To guide the study, we address the following research question:

RQ: To what extent can generative AI tools reduce technical debt in Kotlin code through automated refactoring?

To investigate this research question, we collected data from 46 teams (a total of 19,404 lines of code) and survey responses from each team.

2 Background

2.1 Technical Debt, Refactoring and Generative AI

TD refers to the long-term costs that result from prioritizing short-term delivery or scalability over long-term software quality. TD is often introduced intentionally to achieve rapid delivery, but if left unmanaged, it can make future changes significantly more costly or even impossible. [11, ch. 1]. It can also be introduced unintentionally, for example, by inexperienced teams, or as a consequence of weak development process management [9]. Poorly structured and overly complex code makes changes more difficult and error-prone, increasing the likelihood that developers unintentionally introduce bugs. These issues often reduce developer productivity and make systems more fragile, with the consequences typically becoming apparent first to the development team. [11, ch. 2] In agile contexts, technical debt rarely appears in isolation; it coexists with social and process debt, and these forms reinforce each other over time [15].

Refactoring involves restructuring the internal design of software to enhance understandability and maintainability, while ensuring that the system's

observable behavior remains unchanged. Changes introduced to achieve short-term goals, often without a full understanding of the overall design, gradually weaken the code structure. As a result, the underlying design becomes less visible and harder to maintain. Regular refactoring helps prevent this by preserving structure and improving code readability. [6, ch. 2]. Examples of refactoring techniques include Extract Function, in which a fragment of code is moved into a separate, well-named function that reflects its purpose. By separating intention from implementation, this approach makes the code easier to comprehend and maintain, as readers can understand what the code does without focusing on its internal details [6, ch. 6].

Generative AI tools became widely known to the public after OpenAI launched ChatGPT in late 2022. Generative AI can produce new content such as text, code, images, video, and audio. These capabilities are largely driven by large language models (LLMs) [12]. AI-assisted developer tools can accelerate work, help identify and fix errors, maintain code and documentation consistency, and reduce developers' stress [17]. *Gemini* is Google's generative AI platform based on LLMs. *Gemini* in Android Studio is an AI assistant tailored to Android development that understands natural language, helping developers generate code, learn best practices, document code, and more [8]. *GPT UiO* (hereafter, GPT) is a service from the University of Oslo (UiO) that provides access to OpenAI's GPT models within UiO's privacy and security framework. The data are stored on UiO servers, and inputs/outputs are temporarily processed by Microsoft Azure OpenAI within GDPR-compliant regions, but data are not used to further train external models [19].

2.2 Static Code Analysis and SonarQube

SonarQube is a static code analysis tool that supports more than 30 programming languages and frameworks. It uses predefined, language-specific rules based on coding standards, best practices, security vulnerabilities, and code complexity to detect quality, maintainability, and security issues. The tool can detect issues in seconds that might take a developer hours to locate. It produces reports identifying issues and estimating remediation effort, helping teams to plan for maintainability and address TD earlier [10]. Some metrics and measures include size metrics, code smells, TD estimation and two kinds of complexity.

Code smells harm maintainability by making code harder to understand and modify. They rarely cause runtime errors but signal deeper design problems and increased TD. Examples include large classes and long methods, which raise cognitive complexity and reduce maintainability. Code smells are severity-classified to help prioritize which ones to resolve first [16], from highest to lowest: blocker, critical, major, minor, and info [16].

Cognitive complexity is a metric that measures the difficulty of code for a human to read and the mental effort required to follow its control flow. It can better reflect readability than cyclomatic complexity. For example, two pieces of code with the same cyclomatic score can still differ in readability if one has heavy nesting or convoluted control flow [4].

Cyclomatic complexity measures how complex a program’s control flow is by counting the number of linearly independent execution paths through a function or method. Higher values mean more paths and typically harder to test, harder to understand and harder to maintain. Every method and function scores at least one in the cyclomatic scores [16].

Technical Debt: In SonarQube, TD represents the total estimated effort required to fix code smells. This effort is calculated as the sum of remediation costs for maintainability issues, where each cost corresponds to the estimated time needed to resolve an issue, as defined by the rule that reported it [16].

3 Methodology

We conducted a study in IN2000, a software engineering course at the University of Oslo. The course has served as the empirical setting for several studies of teamwork in agile student teams, including work on gender and collaboration dynamics [18]. The students in this course work on a 10-week project to build a software application that meets specific requirements. Students work in teams to build an Android application using Kotlin and Jetpack Compose in Android Studio. For most students, the course is their first exposure to agile teamwork on a shared codebase. In spring 2025, 343 students enrolled and were organized into 57 teams of five to six students each.

All 57 teams took part in a mandatory refactoring activity, but only 46 consented to research use of their data; we excluded the remaining 11 teams. The 46 included teams contributed 19,404 lines of Kotlin code. Each team selected two files from their own project, one they considered to be of low complexity and one of high complexity, based on their own judgment, with no predefined criteria provided. They used either GPT or Gemini to refactor these files. Before refactoring, teams wrote unit tests for the relevant functionality and verified that they passed. The refactoring procedure had two stages. First, teams sent a general instruction that set the tool’s role and required output format.

Teams then prompted the chosen tool at least three times, using either predefined or self-made prompts. The predefined prompts targeted readability, naming, modularisation, and idiomatic Kotlin style. After refactoring, teams updated their tests to match renamed functions and re-ran them on the refactored code. Each team then completed a post-activity survey on refactoring outcomes, perceived usefulness, and trust in GenAI as a refactoring aid. The survey collected both Likert-scale ratings and free-text responses.

We used SonarQube to quantify the effect of refactoring on maintainability and estimated TD. For each file, we analyzed two variants: the original implementation and the GenAI-refactored version. We deployed a self-hosted SonarQube Community Edition server with default configurations, restricted analysis to Kotlin files, and disabled SCM integration. We organized submitted files into project directories and scanned each project with the SonarScanner CLI. Further analysis details are provided in [1].

Limitations: One limitation of this study is that direct comparison between Gemini and UiO GPT is constrained by the fact that each team submitted different Kotlin files. Although SonarQube analyses were run separately for the two groups, the files varied in size, purpose, complexity, and baseline quality, which can strongly influence which refactorings are possible and how rule violations and TD are reported. These differences may come from the code itself, not from the GenAIs. In addition, the iterative prompting process may have affected results. Students received suggested prompts but could modify them or write their own, so the exact interaction style varied between teams. So our results describe what happened when GenAI tools were used on real student code, but they do not prove that one is better than the other.

4 Results

In this section, we first present results from the student survey, then we present the quantitative results from our analysis with SonarQube.

4.1 Survey Results

Usefulness. The student teams were asked, *“How useful were the AI tool suggestions for improving the maintainability and readability of your code?”*. As seen in Table 1, the student teams reported slightly higher satisfaction with Gemini than with GPT (Gemini: 3.50, GPT: 3.39). This small difference suggests that students perceived both tools as somewhat valuable. One team using Gemini noted: *“Gemini is a useful tool in many ways. It helps a lot when it comes to improving the readability and efficiency of the code. Still, everyone in our group agrees that it only becomes a useful tool if one understands why Gemini does what it does. It was not in all cases that we necessarily understood, even with an explanation, why Gemini’s code was better.”*. Another team using GPT described: *“GPT pointed out some things we had forgotten or not thought of, which was helpful, and some of the new variable names were easier to understand than the originals. Unfortunately, it also produced function and variable names that were worse and harder to understand than the ones we already had.”*

Another team using Gemini was more positive and said *“It contributes to increasing efficiency in the work and most likely ensures cleaner code than one would have managed to produce, and not least had time to produce, on one’s own.”* A team using GPT said *“The code immediately became cleaner and more readable for someone who was not familiar with the codebase. UiO GPT was also useful for commenting and making the code more Kotlin-idiomatic.”*.

Trust. However, although students who used Gemini rated it more valuable, they had lower trust in its results than those who used GPT. One possible explanation is that students may have been more familiar with GPT-like tools because of the popularity of ChatGPT. One student team said (Gemini): *“We both trust and do not trust the AI results. By that we mean that we know AI tools*

Table 1: Student perceptions of AI-assisted refactoring. Likert scale (1 = Very useless; 2 = Slightly useless; 3 = Unsure; 4 = Slightly useful; 5 = Very useful).

Item	Gemini			UiO GPT		
	Mean	Median	<i>n</i>	Mean	Median	<i>n</i>
Usefulness for maintainability and readability	3.50	4.0	23	3.39	4.0	23
Trust in the tool’s results	2.96	3.0	23	3.39	4.0	23
Perceived future usefulness	3.46	4.0	23	4.30	4.0	23

Table 2: Test outcomes after AI-assisted refactorings (% indicate share of teams.)

GenAI	File complexity	All tests passed	Some tests passed	No tests passed
Gemini	Low	87%	4.3%	8.7%
Gemini	High	82.6%	4.3%	13%
GPT	Low	82.6%	8.7%	8.7%
GPT	High	73.9%	13%	13%

can be used as an effective tool, as long as one takes the time to read through and understand why it does what it does - because there are many small errors in the code you get”.

Future usefulness. That said, even though the students found Gemini more useful, when asked about future usefulness, the GPT student teams were more positive. This suggests that even though the tool right now was not as good as they wanted, they had positive feelings about its development. One student team using GPT said: *“We believe that AI can become a useful aid for maintenance and readability over time. It is possible that the technology will improve further in this area. At the same time, we think that larger and more complex programs, such as in this project, may still be challenging for AI to handle on its own.”*

Unit tests. As shown in Table 2, most student teams reported that all unit tests passed after refactoring both files. The rate of fully passing test suites was overall lower for GPT, particularly for the file with high complexity. Some teams experienced partial or complete test failures. These failures were mainly caused by changes in behaviour and functionality, incomplete code fragments, and reference errors. This suggests that GenAI refactoring alone is not sufficient.

4.2 SonarQube Results

Code Smells. We used SonarQube to analyse code smells, and Table 3 shows code smells by severity before and after GenAI-assisted refactoring. Both tools reduced code smells across all severity levels that were affected. Gemini achieved the larger overall reduction (59.8%), with visible decreases in major (44.1%) and minor code smells (69.6%). GPT showed the greatest reduction in critical code

Table 3: Code smells and TD remediation time before and after refactoring.

Severity	Gemini			UiO GPT		
	Before	After	Reduction (%)	Before	After	Reduction (%)
Blocker	0	0	0.0	0	0	0.0
Critical	8	5	37.5	9	2	77.8
Major	34	19	44.1	35	26	25.7
Minor	46	14	69.6	34	19	44.1
Informational	9	1	88.9	9	0	100.0
Total	97	39	59.8	87	47	46.0
TD remediation time (hours)	6.63	3.93	40.72	8.00	5.58	30.25

Table 4: Complexity and size metrics before and after AI assisted refactoring.

Metric	Gemini			UiO GPT		
	Before	After	Change (%)	Before	After	Change (%)
Cyclomatic complexity (CYC)	500	536	7.20	694	838	20.75
Avg. CYC per function	3.11	1.94	-37.62	2.94	1.83	-37.76
Comment lines	440	1027	133.41	472	561	18.86
Functions	161	277	72.05	236	459	94.41
Cognitive complexity (CC)	503	367	-27.06	572	330	-42.38
Classes	34	52	52.94	60	63	5.00
Lines of code (Nloc)	4462	4841	8.50	5090	5011	-1.55

Note: Change is computed as percentage change from before to after refactoring. Positive values indicate increases and negative values indicate decreases.

smells (77.8%) and eliminated all info code smells (100.0%), but with a smaller overall reduction (46.0%).

Technical Debt. As shown in Table 3, SonarQube’s estimated time to remediate TD before and after GenAI-assisted refactorings is reported for Gemini and GPT. Gemini achieved a 10 percentage-point greater reduction than GPT. Both tools reduced the estimated remediation effort, with Gemini performing slightly better.

Complexity and Size Metrics. Table 4 summarizes complexity and size metrics before and after AI-assisted refactoring. Both tools reduced cognitive complexity, with a larger decrease for GPT (-42.38%) than for Gemini (-27.06%). Further, both GenAIs markedly increased the number of functions, suggesting increased modularization, most likely as a result of extract function refactoring technique, while cyclomatic complexity increased for both tools. Gemini differed from GPT mainly in the number of comments and classes, with a large increase in comment lines and the number of classes. Changes in total lines of code were modest for both tools, a slight increase for Gemini and a slight decrease for GPT.

5 Discussion and Conclusion

Our study suggests that GenAI can measurably reduce TD and code smells in agile student code projects. The refactorings reduced cognitive complexity and increased the number of functions and classes, which indicates increased modularity. Both Gemini and GPT reduced code smells and estimated TD hours. These results indicate that GenAI-assisted refactoring can accelerate the resolution of maintainability issues and reduce the estimated remediation time for TD, as captured by SonarQube. Gemini’s overall reduction in code smells was the largest, but GPT had the largest reduction in critical code smells. It is more important to resolve critical issues compared to major, minor, and info issues, where critical issues should be fixed as soon as possible [16]. This suggests that the GenAIs could have different strengths.

Another interesting finding is that total CYC increased after refactoring for both Gemini and GPT, while CC decreased significantly, and the number of functions increased. This result could be explained by, among other refactorings, the refactoring technique “Extract Function”, which splits larger functions into smaller ones with clearer naming, which can enhance code readability and comprehensibility [6, ch. 6], and therefore decrease CC [4]. The increase in total CYC can be explained by the increase in the number of functions, since each function and method has a minimum CYC score of 1, and GPT’s larger increase in total CYC aligns with its proportionally greater increase in the number of functions compared to Gemini. That said, average CYC per function decreased substantially for both tools, suggesting that individual functions became simpler despite the higher total. In practice, the drop in CC suggests the code is easier to read and reason about at the function level [4].

Most teams’ test suites passed, particularly for files of low complexity, but failures were more frequent for GPT code. The failures were mainly caused by behavioral changes, incomplete code, and reference errors. These results show that automated refactoring can improve maintainability and reduce TD, but should be followed by thorough testing and be validated, consistent with prior work highlighting both promise and the need for validation [5, 13].

To conclude, in this study, we examined how Gemini and GPT performed when it comes to code-related TD in agile student teams. Based on an analysis of a survey of 46 teams, we found that students were more satisfied with Gemini but reported greater trust in GPT results. Furthermore, our results suggest that GenAI tools will be valuable tools to reduce TD and improve maintainability in Kotlin code through refactoring in the future, but should be supervised and reviewed by humans, as some did not preserve functionality or produced code that failed to compile. Gemini achieved larger overall reductions within TD and code smells than GPT in our setting. Estimated TD hours were greatly reduced in both GenAI conditions. GPT had the largest reduction in critical and info code smells, but Gemini had the greatest reduction in major and minor code smells.

References

1. Bakke, I.M.: Generative AI-Assisted Refactoring for Technical Debt Reduction and Maintainability: A Case Study of a Software Engineering Capstone Course. Master's thesis, University of Oslo (2026)
2. Besker, T., Martini, A., Bosch, J.: Software developer productivity loss due to technical debt—a replication and extension study examining developers' development work. *J. Syst. Softw.* **156**, 41–61 (2019)
3. Besker, T., Martini, A., Bosch, J.: The use of incentives to promote technical debt management. *Inf. Softw. Technol.* **142**, 106740 (2022)
4. Campbell, G.A.: Cognitive complexity: a new way of measuring understandability. Tech. rep., SonarSource (2023)
5. Cordeiro, J., Noei, S., Zou, Y.: An empirical study on the code refactoring capability of large language models. *ACM Trans. Softw. Eng. Methodol.* (2026). <https://doi.org/10.1145/3801158>
6. Fowler, M.: Refactoring: improving the design of existing code (2018). <https://learning.oreilly.com/library/view/refactoring-improving-the/9780134757681/>. e-book accessed via O'Reilly Online Learning
7. Gehring, J., Kundzich, O., Johnson, P.: AI for mass-scale code refactoring and analysis (2024). <https://learning.oreilly.com/library/view/ai-for-mass-scale/9781098175849/>
8. Google: About Gemini in Android Studio. <https://developer.android.com/studio/gemini/overview> (2025)
9. Irgens, M., Orucevic, M., Boye, M., Gundelsby, J.H., Martini, A.: DebtQuest: discover technical debt management issues with survey visualization. In: 2025 IEEE Working Conference on Software Visualization, pp. 25–35. IEEE (2025)
10. Kahl, I.: Static code analysis: Developer's guide. <https://www.sonarsource.com/resources/library/static-code-analysis/#sonar-and-static-code-analysis>
11. Kruchten, P., Ozkaya, I.: Managing technical debt: reducing friction in software development. Addison-Wesley Professional (2019)
12. Lee, R.: Large Language Models (LLMs) and Generative Artificial Intelligence (GenAI). In: Natural Language Processing, pp. 241–273. Springer, Singapore (2025). https://doi.org/10.1007/978-981-96-3208-4_10
13. Liu, B., Jiang, Y., Zhang, Y., Niu, N., Li, G., Liu, H.: An empirical study on the potential of LLMs in automated software refactoring (2024). <https://arxiv.org/abs/2411.04444>
14. Martini, A., Besker, T., Posch, T., Bosch, J.: TD Pulse: assessing the systematic management of technical debt. In: *IEEE Softw.* **40**(3), pp. 54–62 (2023). <https://doi.org/10.1109/MS.2022.3226035>
15. Martini, A., Stray, V., Besker, T., Moe, N.B., Bosch, J.: Process debt: Definition, risks, and management. *J. Softw. Evol. Process* **37**(4), e70017 (2025). <https://doi.org/10.1002/smr.70017>
16. SonarSource: Understanding measures and metrics. <https://docs.sonarsource.com/sonarqube-server/user-guide/code-metrics/metrics-definition>. Accessed 15 May 2026
17. Stray, V., Moe, N.B., Ganeshan, N., Kobbenes, S.: Generative AI and developer workflows: how GitHub Copilot and ChatGPT influence solo and pair programming. In: Proceedings of the 58th Hawaii International Conference on System Sciences (HICSS) (2025). <https://doi.org/10.24251/HICSS.2025.883>

18. Stray, V., Stabell, I., Sæter, G.E., Barbala, A., Lindsjørn, Y.: Teamwork in agile software development: a mixed-method study of gender diversity and collaboration dynamics. *Inf. Softw. Technol.* **187**, 107840 (2025). <https://doi.org/10.1016/j.infsof.2025.107840>
19. University of Oslo: GPT UiO – UiO’s privacy-friendly GPT chat (2025). <https://www.uio.no/english/services/it/ai/gpt-uio/index.html>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter’s Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter’s Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



**The Fourth International Workshop
on Global and Hybrid Work in Software
Engineering (GoHyb)**



Exploring the Role of AI in Hybrid Work Within Software Engineering

Fateme Broomandi^(✉)  and Maria Paasivaara 

LUT University, Lahti, Finland
{fateme.broomandi,maria.paasivaara}@lut.fi

Abstract. This paper reports insights from the *4th International Workshop on Global and Hybrid Work in Software Engineering* (GoHyb 2026), in which workshop participants, including Agile researchers and practitioners, explored positive and negative future scenarios of AI-supported hybrid and remote work. Our analysis of the discussion shows that participants viewed AI as especially consequential for hybrid work because it can support task review, synthesis, summarization, and asynchronous coordination across distributed teams. At the same time, they expressed concerns that increased AI mediation may weaken human communication, reduce critical discussion, and undermine psychological safety and social connection. Overall, the findings suggest that the central challenge is not simply adopting AI tools, but shaping human–AI collaboration in ways that strengthen coordination without endangering the human capabilities on which software teams rely. The paper closes with implications for future research on AI-enabled hybrid work in software engineering.

Keywords: Human-AI Collaboration · Hybrid Work · Software engineering

1 Introduction

Hybrid work, defined as a work setting in which “*some team members work mostly or completely from home, others mostly or completely from the traditional office, and others in some combination of the two*” [4], has become a defining feature of contemporary software engineering (SE) [15]. While hybrid work in SE offers flexibility, autonomy, and better work-life balance [18], it also introduces persistent challenges related to communication [1], coordination, and maintaining shared awareness [8]. Prior research shows that organizing effective hybrid work in SE depends not only on technical infrastructure, but also on social and relational dynamics such as social and spontaneous interactions [1, 8].

At the same time, generative artificial intelligence (AI) tools have rapidly emerged as a new class of tools being rapidly incorporated into everyday knowledge work, affecting how employees search for information, document work, summarize meetings, coordinate tasks, and make decisions across distributed settings [10]. AI tools help software teams by speeding up development, improving code

quality, and supporting faster learning and knowledge sharing while reducing developers' cognitive load [7]. However, AI in SE reshapes communication and collaboration among developers [17] and significantly impacts communication and collaboration by reducing routine peer interactions, increasing AI-mediated coordination, and requiring new team norms for trust, validation, and shared understanding [7]. These concerns are particularly salient in hybrid work contexts, where communication is already mediated, and opportunities for informal interaction are limited. As a result, generative AI in hybrid work within SE may simultaneously strengthen and weaken key aspects of collaborative work.

In this paper, we summarize the findings from the *4th International Workshop on Global and Hybrid Work in Software Engineering* (GoHyb 2026), held in conjunction with the *27th International Conference on Agile Software Development* (XP 2026). While GoHyb has traditionally examined hybrid work as a socio-technical phenomenon, this year's edition broadened the discussion to include how practitioners perceive AI's role in hybrid work within SE.

2 Related Work

Research on AI in SE has expanded rapidly, showing that generative AI tools are increasingly embedded across SE activities. Hou et al.'s systematic review shows that large language models (LLMs) are now applied across a wide range of SE activities, including requirements engineering, code generation, testing, debugging, maintenance, and documentation [5]. Studies on human-AI collaboration demonstrate that AI supports requirement discussions, Agile meetings, and everyday interactions with coding assistants [2, 16]. Empirical work on coding assistants reports gains in task completion speed and highlights the importance of prompt formulation for code quality [9, 13].

Moreover, Rauf et al. [14] show that AI tools reshape team collaboration and argue that AI should be treated as a team member. A human-AI teaming perspective is suggested, where AI augments but does not replace collaborative and learning processes in SE [14]. At the same time, Treude and Storey [17] argue that AI tools are reshaping developers' roles, transforming software artifacts, and redefining what empirical research must examine. As boundaries blur between humans and machines, artifacts and interactions, and subjects and methods, the empirical SE community must revisit its assumptions, adapt its approaches, and pursue new lines of inquiry [17].

In parallel, research on hybrid work in SE has expanded significantly since COVID-19 [6]. In their systematic literature review, Christensen et al. [3] report persistent challenges faced by software teams working in hybrid environments, including misaligned schedules and work locations, the formation of subgroups, reduced opportunities for informal social interaction, and difficulties in seeking help from teammates. Additional issues affecting alignment and team cohesion arise from unclear information about colleagues' availability and status, miscommunication and misunderstandings, broader communication barriers, and information overload across digital channels [3].

However, these two streams of research have largely evolved separately. There is limited understanding of how generative AI interacts with the challenges of hybrid work in SE. In particular, it remains unclear whether AI mitigates hybrid work in SE issues or introduces new risks for these environments. This paper addresses this gap by examining how participants in the GoHyb 2026 workshop, from academia and industry, perceive the role of generative AI in hybrid work within SE contexts.

3 Method: Interactive Session at the GoHyb 2026 Workshop

GoHyb 2026 took place on the first day of XP 2026, on June 8, in São Paulo, Brazil. It was organized as a half-day, in-person workshop consisting of two main parts. The first part involved 19 participants from academia and industry and included two paper presentations:

- Productivity and collaboration in hybrid agile teams: an interview study [11],
- High trust and high flexibility: hybrid work practices in the Norwegian software industry [12].

To complement these contributions, we presented our ongoing work in the form of a literature review [3], which served to frame the discussion and highlighted current research directions.

The second part of the workshop began with a keynote by Jutta Eckstein, titled *GoHyb Keynote: Global Software Engineering A Retrospective*. A presentation by Xiaofeng Wang, *Support Communication of Virtual Agile Teams: A Metaverse-based Prototype*, followed and helped set the context for subsequent discussions in the interactive part. The workshop ended with an interactive session involving 14 participants.

The interactive session aimed to explore participants' understanding of the role of generative AI in hybrid work within SE. The session explicitly focused on future positive and negative scenarios of AI-supported hybrid and remote work in SE. The session followed the 1–2–4–All structure from Liberating Structures¹. All the participants first reflected individually on either a positive or a negative future scenario. They then discussed their ideas in pairs with another participant who had chosen the same orientation, and two pairs were subsequently merged into a four-person discussion. Finally, the groups shared their main ideas in a plenary discussion. In total, three larger groups reported their views: two groups discussed primarily positive scenarios, and one group discussed negative scenarios. This process allowed participants to first develop individual perspectives and then progressively refine them through collective sensemaking.

The empirical material for this paper consists of two sources produced during the session: participants' post-it notes and a transcript of the plenary discussion.

¹ <https://www.liberatingstructures.com>.

The post-it notes capture short scenario statements, whereas the transcript provides additional explanation, examples, and justifications. Because both sources were organized by discussion group, we could interpret brief written statements in light of the participants' later verbal elaborations. We conducted a qualitative, interpretive analysis to identify recurring themes.

4 Results and Discussion

According to the GoHyb 2026 workshop participants, generative AI is becoming an increasingly influential element of hybrid work in SE. However, their views were not uniform. Participants articulated both positive and negative perspectives on the role of AI in these settings. On the positive side, AI was seen as supporting coordination in distributed and asynchronous work, enabling information synthesis and structuring, and improving productivity in routine and cognitive tasks. On the negative side, participants raised concerns about risks to communication and social interaction, reduced critical thinking and discussion, and growing reliance on AI with implications for trust and accountability.

Despite these differing perspectives, the final plenary discussion revealed a convergence: participants agreed that AI should be integrated into hybrid work in SE practices in ways that support collaboration without undermining human communication and accountability. In particular, they emphasized the need to balance AI support with human collaboration through appropriate guardrails and governance. Figure 1 summarizes these perspectives and their convergence, which we elaborate on in the following sections.

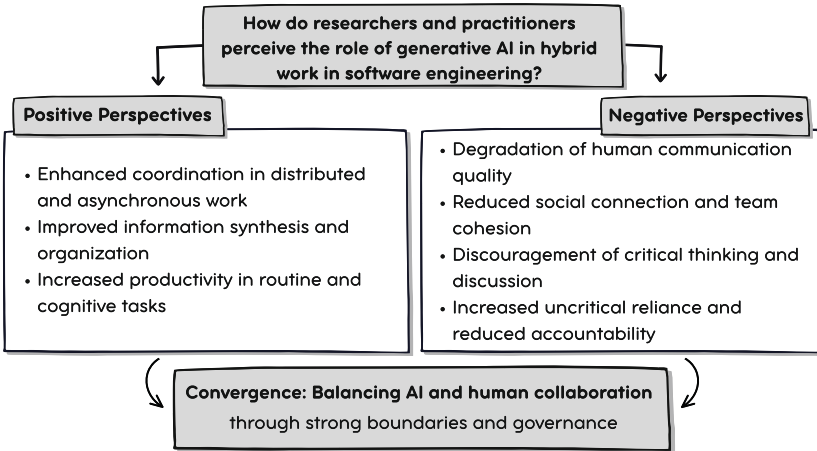


Fig. 1. Perceived role of generative AI in hybrid work in SE.

4.1 Positive Perspectives: AI in Supportive Roles

Workshop participants described several ways in which generative AI can support hybrid work in SE, particularly in relation to coordination, information processing, and productivity.

AI supports coordination in distributed and asynchronous work. According to the participants, the most prominent benefit concerned coordination across distributed teams in hybrid work environments. Participants emphasized that AI can help bridge time-zone differences and asynchronous collaboration by summarizing meetings, tasks, and discussions. For example, one participant noted that *“An AI agent could be useful to summarize tasks or meetings”* for colleagues separated by a *“12-hour distance”*. Similarly, post-it notes highlighted that: *“AI helps ‘sync’ async workers”*, *“AI helps you understand what others are doing”*, and *“AI helps updating other members on tasks”*. These examples suggest that AI aligns well with one of the central challenges of hybrid work in SE: maintaining shared awareness across time and space.

AI helps synthesize and structure information. Beyond coordination, participants highlighted AI’s ability to structure complex information. AI was seen as useful for reviewing content, synthesizing discussions, and organizing scattered knowledge into more accessible forms. This capability was particularly valuable in hybrid work in SE environments, where work is distributed and sometimes overloaded across multiple tools, conversations, and artifacts.

AI improves productivity in routine and cognitive tasks. Participants also associated AI with productivity gains, especially in routine or cognitively repetitive work. AI was described as being able to *“Do boring tasks”*, assist with email handling, generate draft responses, help identify *“Blind spots”*, and improve prioritization and planning. In hybrid work settings, where developers often work asynchronously with fewer opportunities for immediate peer support, participants viewed AI assistance as particularly valuable for maintaining workflow continuity and reducing coordination delays. Importantly, these benefits were not framed solely in terms of individual efficiency. Instead, participants connected them to broader team-level outcomes, such as reducing coordination overhead and improving alignment among distributed actors.

However, participants emphasized that these benefits are conditional, and some of the participants considered AI to be the most appropriate for support tasks such as summarization and structuring, while humans remain central and responsible for judgment and oversight. As one participant noted, the *“Safest task that it can do without needing that much human in the middle is to summarize tasks.”* However, other participants speculated about broader future roles for AI in software work, such as *“One-person teams with agents”*, where a single human collaborates with multiple AI agents. These speculative scenarios further illustrate that some participants viewed AI not only as a support tool but also as a force that could reshape how work is organized and potentially challenge the role of team-based collaboration.

4.2 Negative Perspectives: AI Introducing Risks

The workshop participants highlighted several risks associated with the use of generative AI in hybrid work in SE, particularly in relation to communication and social interaction, critical engagement, and accountability.

AI may degrade the quality of human communication. The most prominent concern was not technical unreliability, but the potential deterioration of direct human communication. Participants worried that if hybrid workers increasingly interact with AI instead of colleagues, they may lose opportunities to practice the messy, corrective, and emotionally grounded aspects of human communication. Participants explicitly argued that human communication is *“flawed, it’s messy, we make mistakes, we apologize for them. There’s kind of a beauty in that”*. Participants emphasized that human interaction, while messy and imperfect, is essential for building understanding, trust, and accountability. While AI may tell people that *“You’re right,”* and eventually *“AI is harming our communication skills”*, *“AI is harming the humanity of our work environments”*, and teams may be *“losing effective communication habits and skills”* over time.

AI may reduce social connection and team cohesion. These concerns were linked to loneliness, culture, and the social texture of hybrid work. One scenario in the post-it notes described *“A team member asking all questions from the AI and at the end of the day feels very lonely as he realizes not talking to a real person during the whole day”*. Participants suggested that such a pattern could deepen withdrawal in remote or hybrid settings and weaken the interpersonal ties that support team cohesion. The discussion also connected this issue to culture and authenticity: participants doubted that AI can meaningfully replace human roles that depend on empathy, emotional understanding, and the sense that another person is genuinely listening rather than only predicting a response. One participant summarized this concern by saying that if people stop practicing human interaction and *“Just talk to machines, it will become every day worse in communication with humans”*.

AI may discourage critical thinking and discussion. Another recurring concern was uncritical reliance on AI-generated outputs. Participants noted that AI systems tend to agree with users, which may limit exposure to alternative perspectives and discourage questioning. It’s particularly problematic in hybrid work within SE environments, where communication and decision-making are already mediated through digital tools. In SE, where quality depends on review, dissent, and clarification, this raises the risk that teams may accept AI-generated outputs without sufficient scrutiny. As one participant stated, *“AI tends to agree with people, so don’t have genuine/critical conversations”*.

AI may increase uncritical reliance and reduce accountability. Closely related to the previous one, participants warned that teams may begin to trust AI outputs without verification, assuming they are correct. This could shift responsibility away from individuals and reduce accountability for decisions. In hybrid

environments, where coordination already depends on mediated interaction, such reliance may further obscure ownership and decision-making processes.

So, the negative scenarios therefore reinforce the argument that the effects of generative AI in hybrid work in SE should be evaluated not only in terms of efficiency, but also in terms of trust, accountability, and the preservation of human collaborative capability.

4.3 Convergence: Balancing AI and Human Collaboration

Across the positive and negative scenarios, participants converged on the idea that the key issue is not simply whether AI brings benefits or risks to hybrid work in SE, nor whether AI should be accepted or rejected in these environments, but rather how AI should be integrated into and bounded within collaborative work practices. Participants consistently described AI as beneficial when it supports coordination, information synthesis, and asynchronous collaboration, but problematic when it replaces human communication, critical reflection, or accountability. In this sense, the discussions pointed toward a middle ground in which AI augments collaborative work rather than substitutes for essential human interaction.

Balancing AI and human collaboration requires strong boundaries and governance. This tension emerged in several ways throughout the discussions. The participants argued that AI should review and synthesize information rather than replace human judgment; AI may assist distributed coordination, but people must remain accountable for interpreting and acting on its outputs; and the more central AI becomes in daily work, the more important it is to preserve direct human interaction. Even the positive notes framed the issue conditionally, stating that *“AI will play a positive role only if we build really strong guardrails and become able to enforce them”*. Participants argued that strong guardrails and the ability to enforce them are necessary if AI is to be integrated into hybrid work in SE without undermining psychological safety.

Taken together, the findings suggest that participants viewed generative AI as a significant and distinctive feature of hybrid work in SE precisely because it operates at the intersection of technical artifacts and human coordination. The positive scenarios point to gains in synthesis, visibility, and asynchronous alignment, while the negative scenarios point to risks for communication, critical engagement, and social connection. Overall, the results indicate that evaluating AI in hybrid work in SE requires particular attention to human capabilities that sustain collaborative SE.

5 Final Remarks

Hybrid work in SE is increasingly shaped by collaboration with generative AI. The findings from the GoHyb 2026 workshop suggest that participants viewed AI as increasingly embedded in the coordination and information practices of hybrid work rather than merely as a peripheral productivity tool. AI was seen as

supporting shared awareness, asynchronous collaboration, and the organization of distributed information across software teams in hybrid work settings. At the same time, participants emphasized that these benefits depend on maintaining human oversight, communication, and accountability. While AI may support coordination, synthesis, and productivity, concerns were raised that excessive reliance on AI could weaken communication, social connection, critical engagement, and collective responsibility. The findings therefore suggest that the role of AI in hybrid work within SE should be understood not only in terms of efficiency gains, but also in relation to the human and collaborative capabilities that sustain software teamwork.

The central implication arising from this workshop is that the value of AI depends on how its role is bounded. Rather than replacing human collaboration, AI must be integrated in ways that preserve communication, trust, and collective responsibility. This requires not only technical capabilities but also clear governance, enforceable guardrails, and evolving team practices.

Future research should not evaluate AI in hybrid work only through speed or convenience. It should also examine how AI adoption affects trust, critical dialogue, loneliness, mentoring, and the human practices through which software teams sustain collaboration. The final implication is therefore about design and governance. Rather than asking whether software teams should simply accept or reject AI, future work should investigate how human–AI collaboration can be bounded, reviewed, and made accountable in hybrid work settings within SE.

Acknowledgements. We would like to thank all participants of the GoHyb 2026 workshop for their valuable contributions and discussions. We especially thank Xiaofeng Wang for organizing the interactive section of the workshop and helping us to analyze the results.

References

1. Ågren, P., Knoph, E., Berntsson Svensson, R.: Agile software development one year into the COVID-19 pandemic. *Empir. Softw. Eng.* **27**(6), 121 (2022). <https://doi.org/10.1007/s10664-022-10176-9>
2. Cabrero-Daniel, B., Herda, T., Pichler, V., Eder, M.: Exploring human-ai collaboration in agile: customised LLM meeting assistants. In: *International Conference on Agile Software Development*, pp. 163–178. Springer (2024). https://doi.org/10.1007/978-3-031-61154-4_11
3. Christensen, E.L., Khanna, D., Broomandi, F., Paasivaara, M., Deshpande, A.: Revisiting hybrid work in agile software engineering: a systematic literature review. Manuscript under review at the *Journal of Systems and Software* (2026)
4. Conboy, K., Moe, N.B., Stray, V., Gundelsby, J.H.: The future of hybrid software development: challenging current assumptions. *IEEE Softw.* **40**(2), 26–33 (2023). <https://doi.org/10.1109/MS.2022.3230449>
5. Hou, X., et al.: Large language models for software engineering: a systematic literature review. *ACM Trans. Softw. Eng. Methodol.* **33**(8) (2024). <https://doi.org/10.1145/3695988>

6. Khanna, D., Christensen, E.L., Gosu, S., Wang, X., Paasivaara, M.: Hybrid work meets agile software development: a systematic mapping study. In: 2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE 2024), pp. 57–67 (2024). <https://doi.org/10.1145/3641822.3641863>
7. Konda, R.: Human-ai collaboration in software teams: evaluating productivity, quality, and knowledge transfer with agentic and LLM-based tools. *BigData, Computational and Management Studies*, 250–257 (2026). *International Journal of AI*
8. Liu, Z., Stray, V., Sporse, T.: Organizational debt in large-scale hybrid agile software development: a case study on coordination mechanisms. In: *Agile Processes in Software Engineering and Extreme Programming – Workshops (XP 2022, XP 2023)*, pp. 75–84 (2024). https://doi.org/10.1007/978-3-031-48550-3_8
9. Mastropaolo, A., et al.: On the robustness of code generation techniques: an empirical study on github copilot. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1887–1899. IEEE (2023). <https://doi.org/10.1109/ICSE48619.2023.00181>
10. Microsoft, LinkedIn: Ai at work is here. now comes the hard part (2024). <https://www.microsoft.com/en-us/worklab/work-trend-index/ai-at-work-is-here-now-comes-the-hard-part>, work Trend Index Annual Report
11. Mo, E., Molléri, J.S., Fagerstrøm, A.: Productivity and collaboration in hybrid agile teams: an interview study. In: *Agile Processes in Software Engineering and Extreme Programming– Workshops (XP 2026)* (2026)
12. Moe, N.B., Stray, V., Hope, M., Willumsen, M.: High trust and high flexibility: hybrid work practices in the norwegian software industry. In: *Agile Processes in Software Engineering and Extreme Programming– Workshops (XP 2026)* (2026)
13. Peng, S., Kalliamvakou, E., Cihon, P., Demirer, M.: The impact of ai on developer productivity: evidence from github copilot. *arXiv preprint* (2023). <https://doi.org/10.48550/arXiv.2302.06590>
14. Rauf, I., Sharp, H., Lopez, T., Wermelinger, M.: Human-machine teaming and team effectiveness in ai tools for software engineering. In: *Proceedings of the 18th IEEE/ACM International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 75–80. IEEE (2025)
15. Smite, D., Moe, N.B., Hildrum, J., Gonzalez-Huerta, J., Mendez, D.: Work-from-home is here to stay: call for flexibility in post-pandemic work policies. *J. Syst. Softw.* **195**, 111552 (2023). <https://doi.org/10.1016/j.jss.2022.111552>
16. Treude, C., Gerosa, M.A.: How developers interact with ai: a taxonomy of human-ai collaboration in software engineering. *IEEE/ACM Forge*, pp. 236–240 (2025). <https://doi.org/10.1109/Forge66646.2025.00033>
17. Treude, C., Storey, M.A.: Generative ai and empirical software engineering: a paradigm shift. In: 2025 2nd IEEE/ACM International Conference on AI-powered Software (AIware), pp. 233–239 (2025). <https://doi.org/10.1109/AIware69974.2025.00033>
18. Šmite, D., et al.: Half-empty offices in flexible work arrangements: why are employees not returning? In: *Product-Focused Software Process Improvement (PROFES 2022)*, pp. 252–261 (2022). https://doi.org/10.1007/978-3-031-21388-5_18

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





High Trust and High Flexibility: Hybrid Work Practices in the Norwegian Software Industry

Nils Brede Moe¹(✉) , Viktoria Stray^{1,2} , Maria Hope³ ,
and Mathias Willumsen⁴ 

¹ SINTEF, Trondheim, Norway
nilsm@sintef.no

² University of Oslo, Oslo, Norway

³ Entra, Oslo, Norway

⁴ Epicenter, Oslo, Norway

Abstract. This paper presents empirical findings from a survey of 172 IT professionals in Oslo, Norway, working in organizations where agile methods are widely used in both private and public sectors. The study explores how hybrid work is experienced and enacted in a high-trust, flexibility-oriented environment. Our results indicate a strong employee preference for flexible work models that allow autonomy in time and location. Remote work is widely used, sometimes even substituting for sick leave during mild illness. At the same time, the respondents report that office attendance is important for social connections. The data further reveal high levels of trust in colleagues' effectiveness when working remotely, with low concern about misuse of home office privileges. Some organizations have introduced mandatory office days, often with the stated goal of strengthening culture and workplace community. Overall, the hybrid setups are perceived positively and are not seen as a major factor in employee turnover. Instead, career progression and compensation remain the primary drivers of job change. These findings offer insights into how hybrid work practices are integrated into agile ways of working in a Scandinavian context.

Keywords: Collaboration, Coordination, Alignment, Hybrid work, Remote participation, Teamwork, Large-scale agile development.

1 Introduction

Hybrid work has officially transitioned from a temporary pandemic-era response to the “new normal” in software engineering, blending remote and office settings to provide practitioners with enhanced flexibility and a better work-life balance [2, 7]. Now, agile team members often have the flexibility to choose, at least to some degree, between remote and office-based work [4]. Yet, the norms and expectations around hybrid work are still evolving, prompting new questions about how flexibility is interpreted and exercised in software teams [13].

But what does flexibility actually look like in everyday practice? Smite et al. [14] surveyed 20 companies and found that preferences for the proportion of time spent in the office versus at home vary both across and within companies. Moe et al. [11] explored the co-presence patterns of 17 agile teams in a company whose employees work partly from home and found significant variation in co-presence practices. Further, they observed slightly higher attendance rates among women, but these differences were statistically insignificant. Klotins et al. [8], in a survey of two large agile Scandinavian software companies, found that gender differences in office attendance preferences and remote work behavior among software developers were minimal and statistically insignificant. Instead, other variables, such as the degree of collaborative work, commute time, and responsibility for supporting teammates, show a stronger association with both actual and preferred office attendance. The choice between remote and office days also shapes how teams enact specific agile practices. For example, Tkalic et al. [15] found that pair programming in hybrid work can take place as on-site, remote, or mixed sessions, with mixed sessions perceived as the least advantageous, and recommended that teams align office presence to enable either fully on-site or fully remote pairing.

Empirical studies reveal that hybrid work is not a uniform practice, but a negotiated and context-dependent arrangement shaped by organizational norms, team dynamics, and individual circumstances. The top motivations for encouraging or mandating office work are largely rooted in factors that benefit the organization (corporate culture, engagement, innovation) and work groups (collaboration and social relationships) [13]. In contrast, motivations for remote work tend to center on individual-level benefits (work-life balance, focus, and efficiency) [13]. Therefore, hybrid work introduces inherent tensions between individual autonomy and collective cohesion that organizations must navigate.

Hybrid work challenges how teams maintain social connection and belonging. Belonging is primarily built through daily social interactions in agile software development [6]. However, hybrid work reduces the frequency of spontaneous and random office encounters, and organizations must now take continuous, conscious, and visible actions at the individual, team, and organizational levels to maintain the social connections vital for motivation and resilience [2]. Prior research also suggests that communication modality may influence emotional expression in agile teamwork. For example, Awan et al. [1] found that hybrid retrospective meetings in student Scrum teams were associated with more emotionally restrained interactions than fully in-person meetings, highlighting potential challenges of partial presence in distributed collaboration. Trust is another critical condition for hybrid collaboration. In hybrid software development teams, trust has been linked to leadership behavior, organizational policy and appraisal mechanisms, communication practices, individual attributes, and the role of digital technologies, including how monitoring tools are perceived [16].

Balancing the needs of organizations, teams, and individuals remains a central challenge in hybrid work. In response, many companies have begun re-

evaluating their hybrid strategies—ranging from gentle nudges for on-site presence to formal return-to-office mandates. Norway is often characterized as a high-trust work environment with relatively autonomous employer–employee relations, making it an interesting setting for studying hybrid work practices. A recent survey by Smite et al. [13] found that requiring three office days per week is the most common mandate among 68 companies. Right after the pandemic, only 15% had implemented such policies; by 2025, this figure had risen to 26%. To better understand current preferences and the lived experience of hybrid, we ask the following research question: *What are the preferences and experiences of hybrid work among IT professionals, and what factors influence office attendance and perceptions of remote work?*

The study investigates how hybrid work is enacted in practice, what employees value in office versus remote work, and how trust and flexibility relate to perceived effectiveness, attendance norms, and retention intentions.

2 Context and Methodology

To investigate hybrid work practices, we analyzed survey responses from a subset of 172 IT professionals drawn from a larger survey of 1,000 office workers in the Oslo region of Norway. The quantitative data were collected by the market research company Opinion through a web-panel survey conducted during weeks 7–9 in February 2025. The broader sample targeted full-time employees aged 23–65 working primarily in knowledge-intensive office occupations. From this dataset, we extracted respondents who reported working in IT-related roles, which constitute the empirical basis for this paper.

The survey covered themes related to hybrid work practices, including home office usage, flexibility preferences, workplace attendance, trust in remote work, perceptions of workplace culture and leadership, work–life balance, and retention intentions. Most questions were close-ended and used predefined response categories or five-point Likert scales. A limited number of questions also allowed open-ended “other” responses; however, this paper focuses exclusively on the quantitative data. All responses were anonymous and self-reported.

All participants worked in the Norwegian software industry, where agile methodologies are widely used in both private and public sector organizations. Consequently, the respondents largely operate in agile team settings, which form an important backdrop for interpreting the hybrid work results. The sample was not intended to be statistically representative of the entire Norwegian IT workforce, but rather reflects employees in a Scandinavian work environment characterized by relatively high levels of trust and widespread adoption of hybrid and agile work practices.

We primarily used descriptive statistical analyses, including frequencies and percentages, to examine hybrid work preferences, attendance patterns, and attitudes toward office and remote work. The broader workplace study also included a small number of qualitative interviews with practitioners and experts, but the analysis reported here focuses solely on the quantitative findings from the IT subsample.

Table 1. Hybrid work preferences and home office practices (N = 172)

Dimension	Category	Percentage
Work arrangement preference	Flexible time and location	73%
	Fixed hours and location	17%
	Fully integrated work and private life	7%
	Other / undecided	3%
Home office days per week	0 d	30%
	1 day	22%
	2 d	23%
	3 d	14%
	4 d	8%
	5 d	4%

Finally, this study has several limitations. The findings are based on self-reported survey data and therefore reflect respondents’ perceptions rather than directly observed practices. In addition, the study focuses on a single regional setting (the Oslo region), which may limit the generalizability of the findings to other national, cultural, or organizational contexts outside Scandinavia.

3 Results

3.1 Preference for Flexible Work Arrangements

Flexibility emerged as a highly valued aspect of work among the IT workers (Table 1). The majority of the IT respondents reported a preference for a flexible work model where they can adjust their workdays around personal obligations or preferences (e.g. working remotely when suitable), rather than a fixed schedule. Only about one-fifth preferred a traditional arrangement with strictly set hours and a clear separation between work and personal life. Only 7% wanted a fully integrated approach where work and free time blend with no firm boundaries. These results indicate that most employees prioritize flexibility in when and where they work, aligning with agile principles that emphasize autonomy. Moreover, about two-thirds of the respondents reported enjoying a good work–life balance in their current setup, suggesting that the prevailing hybrid arrangements are, on the whole, meeting employees’ needs. Interestingly, IT employees reported the highest number of days working from home across the full sample, while also scoring highest on work–life balance. Overall, the findings underscore a strong employee demand for adaptable work arrangements in the post-pandemic era.

3.2 Social Connection as the Primary Office Draw

Despite the appetite for flexibility, respondents still see clear value in the physical office, chiefly for social reasons. The social connection with colleagues was cited

Table 2. Main reasons for choosing the office over home office (N = 172).

Factor	Percentage
Social connection with colleagues	69%
More effective collaboration across teams	42%
Contributing to a better office culture	42%
I learn more from my colleagues	34%
Important for my mental health	26%
Better lunch and food experiences	24%
Better access to effective work equipment	21%
Better work–life balance	19%
I am more visible to my direct manager	17%
Makes me more productive	17%
Easier to focus on work tasks	17%
Better for my career development	11%
Makes me more creative and innovative	5%
None of these or do not know	5%

as the number-one motivator for coming into the office (Table 2). In fact, about two-thirds of the IT professionals indicated that collegial social interaction and camaraderie are the most important reasons to meet in person at work. This finding suggests that the office is viewed less as a place one has to go for productivity or collaboration, and more as a place one wants to go to nurture a sense of community. Other potential factors (such as face-to-face collaboration or access to equipment) were less commonly ranked at the top. For example, building or contributing to a better workplace culture was important, but still mentioned by fewer (around 42%) compared to the social aspect. Notably, agile teams typically emphasize communication and teamwork, and these survey results confirm that in-person social bonds are a key draw even in highly digital, collaborative work environments. In summary, office attendance among the surveyed IT professionals appears to be driven primarily by a desire for social belonging rather than by direct productivity concerns. This interpretation is also consistent with prior organizational research emphasizing the role of office communities in supporting culture and informal knowledge sharing.

3.3 Home Office Usage Patterns and Health-Related Flexibility

The data reveal a predominantly hybrid work pattern among the surveyed IT staff (Table 1). Only about 30% of respondents never work from home at all, meaning they are in the office every working day. The majority (approximately 45%) practice a 1–2 d per week home-office schedule, splitting their time between home and office. Few (26%) report using home office more frequently than two days a week. This indicates that one to two remote days weekly is the common

hybrid mode for IT employees in Oslo Norway. (For context, in the broader cross-industry sample, 41% worked exclusively on-site with 0 remote days, suggesting that IT professionals utilize home office somewhat more often than other sectors.) The prevalence of hybrid work is also reflected in how employees handle minor illness: 62% reported opting to work from home instead of taking sick leave when facing mild illness or discomfort, and 8% reported working from home when their kids were sick. In other words, over two-thirds used home office as an alternative to sick leave within the past year. This practice can be seen as an adaptive benefit of hybrid arrangements, allowing continuity of work during mild colds or situations like a sick child at home, without formally going on sick leave. It may contribute to reducing absenteeism, though it also blurs the line between being unwell and working. Overall, the patterns suggest that hybrid work is well-entrenched: most IT employees mix home and office days, leveraging the flexibility to maintain productivity even when they might previously have had to call in sick.

3.4 Trust, Culture, and Mandatory Office Days

The survey results indicate a high-trust culture around remote work in these agile IT teams. Most respondents reported confidence that their colleagues work as effectively from home as in the office. In particular, 75% selected the two highest response categories when asked about trust in colleagues’ effectiveness while working from home (Table 3). At the same time, concerns about the overuse of the home office were low. Only about one-fifth of respondents reported believing that colleagues use home office too much, while most rejected this view. Taken together, the results suggest that remote work is largely interpreted through norms of trust and accountability, which aligns with agile assumptions about self-management and transparency.

Table 3. Trust that colleagues work as effectively from home as from the office.

Response category	Percentage
1. To a very small extent	3%
2. To a small extent	5%
3. To some extent	16%
4. To a large extent	25%
5. To a very large extent	50%

In response to hybrid work trends, many organizations have recently introduced mandatory in-office days (e.g. requiring attendance on certain weekdays). Notably, these policies are often implemented with the intention to strengthen workplace culture and team environment, rather than to micromanage productivity. In fact, our results indicate that 66% of firms imposing office-day requirements cited “building culture and a good work environment” as the primary

reason, far outweighing motives like monitoring employees. This aligns with the high trust levels – voluntary and trust-based approaches seem more effective than rigid mandates in the Norwegian agile context. In summary, organizations are trying to find a balance: encouraging physical co-presence to foster culture, without undermining the valued flexibility and trust that hybrid-agile workers cherish.

3.5 Job Satisfaction, Retention Intentions, and Workplace Quality

The IT respondents generally appear satisfied with their current hybrid work setups. Hybrid work itself is not a major source of job dissatisfaction – on the contrary, the flexibility is often seen as a plus. When asked about reasons that would drive a job change, work arrangement issues ranked low. The top reasons for changing jobs were instead seeking better compensation and new challenges, consistent with broader industry trends. For example, in the overall sample, desire for higher pay and desire for new challenges were the most common reasons given for the last job move. By contrast, dissatisfaction with the present hybrid/remote work policies did not figure prominently. Additionally, most participants signaled a sense of stability with their employers: half of the IT respondents expected to stay with their current employer for at least the next three years, and only about 21% anticipated leaving sooner (the remaining 24% were unsure or undecided about their tenure). This suggests relatively moderate turnover intentions, indicating that the present work environment, with its hybrid setups, is generally working well for most.

Another finding is the important role of the physical office environment for talent retention and attraction. An overwhelming 80% of respondents said they would consider rejecting a job offer if the prospective employer’s office facilities were of poor quality. In other words, workplace quality matters: even in an age of hybrid work, employees care about having an appealing, well-equipped office to come into. This highlights that organizations cannot focus solely on remote settings; they also need to invest in high-quality office spaces that make the commute “worth it.” Participants indicated that an appealing and functional office environment, one that supports both social interaction and productivity, is a key factor in how they evaluate potential employers

4 Discussion and Conclusion

This study analyzed survey responses from a subset of 172 IT professionals drawn from a larger survey of more than 1,000 office workers in Norway in the Oslo region (capital of Norway). In Norway’s software development industry, agile methodologies are the dominant practice in both private and public sector teams. Our finding that 73% of respondents prefer flexible arrangements reinforces the broader observation that hybrid work is stabilizing as a long-term work mode in software engineering [13], rather than a transitional post-pandemic phase. Broomandi et al. [2] argue that organizations are still searching for “balanced

setups,” and that research must provide evidence-based guidance on hybrid practices and policies rather than relying on intuition. Our findings, therefore, provide quantitative evidence from a Norwegian IT environment, demonstrating that employees’ desire for autonomy in both their schedules and work locations persists, consistent with the broader research agenda on autonomous agile teams that has identified autonomy at both the individual and team level as a defining principle of agile work [9]. This suggests that inflexible policies could be a source of conflict.

When examining the factors that influence office attendance, social connection with colleagues emerged as the strongest motivation for being physically present at the office. This echoes the findings of Smite et al. [12] that found that the most common reason for onsite work is related to connection and community. It also supports that it is important to have meetings requiring active discussions on-site, while more general information sharing meetings for larger audiences can be virtual so participants can attend from home [3].

A defining feature of hybrid work in the Norwegian IT sector appears to be a high degree of interpersonal trust among team members. Most respondents reported confidence that colleagues are as effective when working from home as in the office, and only a minority suspected misuse of home office arrangements. This pattern aligns with prior work that emphasizes trust as a central condition for collaboration in hybrid software development, particularly given reduced informal interaction and potential coordination challenges in hybrid settings [16].

This interpretation resonates with Tyagi and Masood’s argument that trust in hybrid teams is shaped by organizational and leadership factors, including hybrid work policies and the broader organizational culture around fairness and inclusion [16]. In a high trust context, mandatory office days may therefore function less as a control mechanism and more as an attempt to compensate for the loss of spontaneous social interaction that supports team cohesion over time.

Interestingly, the increased use of mandatory office days in many organizations appears, in our data, to be framed primarily as a culture-building measure rather than a response to low trust, or concerns about productivity, innovation and organizational cohesion which have been found in a recent study [5].

While hybrid work arrangements appear to satisfy most employees’ preferences, the quality of the physical office remains a decisive factor in job evaluations. It seems that in Norway, there is a good balance between what employees and employers want. However, 80% reported that they would consider rejecting a job offer if the office facilities were of poor quality. In an earlier study [10], we found that office quality was one of the workplace attributes engineers weigh when choosing where to work.

Overall, the findings suggest that hybrid work experiences in the Norwegian software industry are shaped by a combination of high trust, employee autonomy, and the continued importance of the office as a social and cultural meeting place.

Acknowledgements. The work was partially supported by the Research Council of Norway through the project Vertigo (grant 346563).

References

1. Awan, W.N., Paasivaara, M., Gloor, P.A.: When emotions matter: speech-based emotion detection in capstone project retrospective meetings while working hybrid. In: International Conference on Software Engineering (ICSE-SEET'26), pp. 12–18 (2026)
2. Broomandi, F., Paasivaara, M., Christensen, E.L., Hyrnsalmi, S., Khanna, D.: Hybrid work in agile software engineering: Current research and future directions. In: Peter, S., et al (eds.) Agile processes in software engineering and extreme programming - workshops, pp. 53–67. Springer Nature Switzerland, Cham (2026)
3. Christensen, E.L., Paasivaara, M., Salman, I.: Hybrid work in agile software development: recurring meetings. In: 2025 IEEE/ACM 18th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE), pp. 120–130. IEEE (2025)
4. 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). <https://doi.org/10.1109/MS.2022.3230449>
5. Eng, I., Aboagye, E., Bergsten, E.L., Strömberg, A.: Balancing employee flexibility and organizational performance: implications for innovation, productivity, and company attractiveness in smes. *Front. Psychol.* **16**, 1518284 (2025)
6. Hyrnsalmi, S.M., Broomandi, F., Salman, I., Paasivaara, M.: Fostering a sense of belonging in hybrid work within agile software development. In: International Conference on Agile Software Development, pp. 37–51. Springer (2025)
7. Khanna, D., Christensen, E.L., Gosu, S., Wang, X., Paasivaara, M.: Hybrid work meets agile software development: a systematic mapping study. In: IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE '24) (2024). <https://doi.org/10.1145/3641822.3641863>
8. Klotins, E., Smite, D., Chatzipetrou, P., Tklich, A., Brede Moe, N.: Interest in working remotely: what factors are at play? *J. Softw. Evol. Process* **38**(2), e70084 (2026). <https://doi.org/10.1002/smr.70084>. <https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.70084>
9. 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: Agile Processes in Software Engineering and Extreme Programming- Proceedings of the XP 2019 Workshops, pp. 13–19. Springer (2019)
10. Moe, N.B., Stray, V., Smite, D., Mikalsen, M.: Attractive workplaces: what are engineers looking for? *IEEE Softw.* (2023). <https://doi.org/10.1109/MS.2023.3276929>
11. Moe, N.B., Ulsaker, S., Hildrum, J.M., Smite, D., Ay, F.C.: Understanding the difference between office presence and co-presence in team member interactions. In: Proceedings of the 57th HICSS. AIS Electronic Library (2024)
12. Šmite, D., Klotins, E., Moe, N.B.: What attracts employees to work on site in times of increased remote working? *IEEE Softw.* **42**(1), 100–109 (2024)
13. Smite, D., et al.: Who “controls” where work shall be done? state-of-practice in post-pandemic remote work regulation (2025). [arXiv:2505.15743](https://arxiv.org/abs/2505.15743) arXiv preprint

14. Smite, D., Moe, N.B., Hildrum, J., Gonzalez-Huerta, J., Mendez, D.: Work-from-home is here to stay: call for flexibility in post-pandemic work policies. *J. Syst. Softw.* **195**, 111552 (2023)
15. Tkalic, A., Moe, N.B., Andersen, N.H., Stray, V., Barbala, A.M.: Pair programming practiced in hybrid work. In: *International Symposium on Empirical Software Engineering and Measurement (2023) ESEM*
16. Tyagi, S., Masood, Z.: Understanding factors influencing trust in software development teams in hybrid work settings: an empirical investigation. In: *International Conference on Agile Software Development*, pp. 78–87. Springer (2025)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Productivity and Collaboration in Hybrid Agile Teams: An Interview Study

Elisabeth Mo^{1,2}, Jefferson Seide Molléri²(✉) , and Asle Fagerstrøm²

¹ DNB Bank ASA, Oslo, Norway

`elisabeth_mo@live.no`

² Kristiania University of Applied Sciences, Oslo, Norway

`{jefferson.molleri,asle.fagerstrom}@kristiania.no`

`http://www.kristiania.no` , `http://www.dnb.no/`

Abstract. Hybrid work has become a reality post-pandemic, transforming how Agile teams deliver value, collaborate, and adapt. This study investigates how hybrid settings influence productivity and collaboration through nine interviews with three Norwegian Agile teams. Our findings show that hybrid work reduces informal interaction, creates uneven participation, and increases reliance on digital tools. Agile ceremonies became alignment anchors, while trust, communication, and tool support mediate team effectiveness. Hybrid Agile work is an evolving field that requires tailored structures to support inclusion, team cohesion, and sustainable performance.

Keywords: Hybrid Agile teams · remote participation · team inclusion · collaboration · productivity · sustainable Agile

1 Introduction

Hybrid work has become a feature of the modern workplace, with many co-located teams now operating in between digital and physical spaces. Several years post-pandemic, organizations embraced flexibility and autonomy, yet teams continue to struggle with alignment and cohesion across distance [11, 12, 14]. These tensions are particularly visible in Agile environments, which emphasize close collaboration, knowledge sharing, and effective task coordination [2, 4].

Hybrid settings disrupt the conditions under which Agile teams work due to scheduling conflicts, miscommunication, and a weakened sense of team cohesion [3, 6]. Beyond that, it raises concerns related to sustainable pace, psychological well-being, and equal participation in decision-making [6, 15].

This study explores these challenges through a qualitative investigation of three Agile teams from Norwegian public and private organizations. We address the following research question: *How do hybrid work environments influence productivity and collaboration within Agile teams?* Our contribution is an empirical account of how Agile practices are being adapted to sustain inclusion, well-being, and effectiveness in hybrid settings.

2 Background

Agile methods were originally built around face-to-face interaction, short feedback loops, and collective ownership of work [2,4]. In hybrid work, these foundations are increasingly strained. Informal communication and spontaneous coordination become harder to sustain [9], while digital tools intended to bridge distance can introduce friction when used inconsistently, leading to fragmented awareness and slower decisions [3].

Hybrid work blends remote flexibility with periodic office presence [6], offering autonomy and worklife integration but complicating Agile’s reliance on continuous collaboration. Remote members may feel excluded when decisions occur among co-located colleagues [12], reinforcing the need for equal access to information, and support for diverse working rhythms [8]. Hybrid settings expand talent pools and enable flexible scheduling, but require deliberate coordination structures to maintain team cohesion and ensure inclusion [1,11,14].

Hybrid conditions directly affect *productivity* (how efficiently teams deliver value) and *collaboration* (their ability to share knowledge, coordinate work, and maintain alignment across locations). Reduced communication bandwidth limits tacit knowledge sharing and non-verbal cues that foster trust and shared understanding [1]. Participation in Agile ceremonies can become uneven, with remote members struggling to contribute to discussions dominated by those on-site [6,12]. Although tools such as Slack, Teams, and Jira are essential for distributed coordination, inconsistent usage can delay feedback and disrupt workflow [3,9].

3 Method

Data Collection. Data were collected in 2025, when hybrid work practices had stabilized post-pandemic. The study involved three Agile teams in Norway: two from the financial sector (DNB - *Den Norske Bank*) and one from the public sector (*Digitaliseringsdirektoratet*).

Participants included software engineers, developers, product owners, and managers, ranging from early-career to experienced (10+ years) professionals. Teams varied in size, roles, and Agile maturity, and worked in cross-functional setups. All teams operated in partially co-located arrangements with regular office days used for Agile ceremonies (such as sprint planning), and collaborative sessions, while development work largely occurred remotely. This setting reflects common hybrid Agile practices in contemporary organizations.

We conducted nine semi-structured interviews with practitioners via Microsoft Teams in March 2025. Interviews lasted 45–70 min, and were recorded with participant consent. To ensure confidentiality, all statements were anonymized. Participants received the interview guide in advance.

The IMOI framework allowed us to explore how team dynamics unfold through *inputs* (e.g., hybrid work conditions, tool use), *mediators* (communication quality, shared awareness), *outcomes* (productivity and coordination effectiveness), and *impacts* (adaptability, sustainability, and well-being). These codes

were then grouped into higher-level themes, which form the basis of the results in Sect. 4 (Table 1).

Table 1. Interview quotes and their corresponding assigned codes.

Interview Quote	Assigned Codes
<i>“When we’re all working remotely, it’s harder to read the room during planning meetings. Sometimes, I can’t tell if the team actually agrees or if they’re just quiet”.</i>	<div>- <i>Input</i>: remote communication challenges</div> <div>- <i>Input</i>: lack of non-verbal cues</div> <div>- <i>Mediator</i>: unclear team alignment</div>
<i>“Some developers prefer working late at night, while others during the day. That can sometimes delay feedback and reviews”.</i>	<div>- <i>Input</i>: timing misalignment</div> <div>- <i>Input</i>: asynchronous work challenges</div> <div>- <i>Mediator</i>: impact on feedback</div>
<i>“Having one or two set days in the office makes a difference. We use that time for workshops or tougher decisions that are hard to do efficiently online”.</i>	<div>- <i>Input</i>: hybrid schedule structure</div> <div>- <i>Mediator</i>: office time for high-impact tasks</div> <div>- <i>Outcome</i>: in-person decision-making</div>

4 Results

4.1 Collaboration in Hybrid Agile

Participants emphasized that hybrid work has transformed how Agile teams collaborate. Without the spontaneous interactions of co-located work, teams depend more on structured communication, leading to gaps in shared awareness and uneven participation. One participant explained: *“Sometimes, I just hear about a small decision in passing, and it turns out it was made while I was working from home”*. Collaboration challenges reflect *mediators* such as weakened team processes that are influenced by hybrid *inputs* like asynchronous availability. The main findings in this theme are:

- Hybrid setups created gaps in information flow when on-site discussions were not shared digitally.
- Office-based decisions may exclude remote members, causing misalignment.
- Fewer informal interactions limited clarifications and relationship-building.
- Effective collaboration depended on team culture, including trust, openness, and proactive communication.
- Agile ceremonies served as alignment anchors, compensating for the loss of informal coordination.
- Retrospectives are safe spaces for honest feedback and addressing tensions.

4.2 Communication and Tool Use

Digital tools emerged as *inputs*, both enablers and constraints for hybrid Agile work. Teams rely on Slack, Microsoft Teams, and Jira to sustain transparency and asynchronous coordination, yet inconsistent use and unclear conventions frequently hinder efficiency. As one interviewee pointed out: “*We often assume something has been communicated, but in reality, it depends on who was in the room or which channel it was posted in*”. Communication tools are *inputs* that influence *mediators* such as coordination and shared understanding.

- Digital tools help bridge distance and support asynchronous coordination.
- Async updates supported flexibility and protected focus time.
- Inconsistent communication routines led to missed updates and uneven engagement within teams.
- Misalignment in communication expectations created confusion about urgency, intent, and decision status.
- Remote collaboration made work handoff more difficult, reducing shared understanding between roles, e.g. designers and developers.

4.3 Adaptability and Agile Mindset

Adaptability emerged as a defining *impact* for teams that thrived in hybrid settings. Yet, this adaptability was uneven across members and often shaped by organizational culture. It reflects a key *mediator*, that mean, an emergent team process through which hybrid work *inputs* are managed. An interviewee noted: “*We spend a lot of time in meetings. Sometimes I wish we had more blocks of quiet time to actually do the work.*” The main codes identified are:

- Sprint planning provides direction, but teams often adjust priorities mid-sprint to accommodate shifting availability in hybrid work.
- Variation in Agile maturity affected coordination, with some members applying Agile principles more consistently than others.
- Greater autonomy enable teams to adapt more effectively to hybrid by re-aligning work without waiting for managerial decisions.

4.4 Productivity and Performance Drivers

Participants linked productivity to disciplined Agile practice and mindset. The hybrid setting introduced both enablers and barriers to maintaining flow. One participant explained: “*We’re most effective when we’re trusted to make the call and adjust as needed.*” The productivity *outcomes* described here emerge from earlier relationships between *inputs and mediators* such as work arrangement → coordination and digital tools → communication. The main codes identified are:

- Sprint planning helped hybrid teams set shared goals and identify work dependencies.

- Frequent digital meetings interrupted work needed for development tasks.
- Dependencies on remote actors from outside the Agile team caused delivery delays.

4.5 WorkLife Balance and Satisfaction

Hybrid work also reshaped how participants experienced motivation, well-being, and sustainable performance. These effects represent an *impact* with broader consequences for individuals beyond immediate team *outcomes*, and they may cycle back to influence future *inputs* such as availability and participation. As noted by a participant: “*We’re technically allowed to work from home, but it’s kind of expected that you show up a few days a week no one says it, but it’s there.*” The main topics discussed are:

- Clear worklife boundaries were respected in some teams, supporting balance and reducing burnout risk.
- Unclear expectations to be physically present create subtle pressure despite hybrid policies.
- Agile practices, such as short iterations and limiting work-in-progress, support a sustainable pace under increased coordination demands.
- Learning became more self-driven in hybrid settings, as opportunities for informal knowledge transfer decreased.

5 Discussion

Our findings show that Agile practices are not simply transferred into hybrid settings but adapted through ongoing experimentation. Teams reconfigure ceremonies and communication norms to sustain collaboration and productivity across physical and digital spaces.

Interpreting the findings through the IMOI framework [7] shows how hybrid work reshapes core *inputs* (communication channels, routines, and degrees of co-location) creating both flexibility and new sources of friction. While digital tools support visibility and asynchronous work, decisions made informally in the office can unintentionally exclude remote members, leading to misalignment.

These inputs also shaped *mediators* such as communication quality, trust, and cohesion. Agile ceremonies served as crucial alignment mechanisms that helped compensate for reduced informal contact, yet their effectiveness depended heavily on team culture. Teams with strong norms of openness and proactive communication navigated hybrid constraints more effectively than teams with inconsistent tool use or unclear expectations.

The effects appeared in team *outcomes* and longer-term *impacts*. Productivity and adaptability remained high when teams maintained explicit communication norms and used ceremonies intentionally, but were hindered by external dependencies, meeting overload, or uneven Agile maturity. Sustainable hybrid

work required balancing autonomy with inclusive decision-making and clear worklife boundaries. Teams that practiced transparency and respected personal time reported greater engagement.

5.1 Practical Implications

Our findings support actionable recommendations for strengthen hybrid Agile:

- *Establish intentional communication policies*: define how decisions are shared, how tools are used, and how remote team members are included in informal discussions.
- *Promote decentralized decision-making*: allow teams to act autonomously while ensuring alignment through shared goals and artifacts.
- *Balance ceremonies with focus time*: limit synchronous meetings to essential coordination and use asynchronous updates to preserve deep work.
- *Invest in trust-building initiatives*: use retrospectives and team check-ins to express concerns, celebrate progress, and reinforce psychological safety.
- *Support sustainable pacing*: encourage boundaries between work and personal life and ensure leaders to be exemplars of these behaviors.

These recommendations apply to both team leaders and organizations. Leaders can use them to frame discussions on shared norms rather than impose top-down rules, while organizations must ensure inclusive and flexible participation. Hybrid agility succeeds when individuals, teams, and structures evolve together within a coherent ecosystem.

5.2 Comparison with Related Work

Our findings align with existing research on hybrid Agile work while offering a more mature perspective compared to early pandemic studies (e.g., [13,16]). By 2025, hybrid work had become a stable and accepted model, and participants described persistent practices (such as hybrid schedules and implicit presence expectations) rather than temporary adaptations.

Consistent with prior work [12,13,16], we found that hybrid settings reduce informal interactions and create risks of excluding remote members. Our results extend this by clarifying how such gaps arise, for example through micro-decisions or ad-hoc clarifications occurring in the office. Literature emphasizes the need for deliberate communication structures [12]; our study reinforces this and shows that Agile ceremonies act as alignment anchors, though their effectiveness depends strongly on trust, openness, and shared norms.

We further contribute by highlighting how inclusion and trust mediate the link between communication quality and productivity - dimensions underexplored in previous studies. We also identify challenges such as ceremony overload, external dependencies, and uneven Agile maturity, which influence hybrid effectiveness but are rarely discussed. Finally, our findings extend the notion of sustainability beyond job satisfaction [8,16], showing how hybrid work amplifies Agile’s “sustainable pace” principle by shaping long-term team cohesion and well-being.

6 Limitations and Threats to Validity

The main limitations of our study (according to [10]) are:

Credibility may be affected by the use of self-reported experiences and researcher interpretation. We mitigated this through triangulation across teams, and iterative checks against the IMOI framework.

Transferability is strongest for mature hybrid settings, as data were collected after hybrid work had stabilized. While convenience sampling may limit breadth, we included participants with varied roles and experience levels.

Dependability was supported by using a semi-structured interview guide, applying consistent procedures across interviews, and documenting coding steps. Categories were refined iteratively to ensure a traceable analytic process.

Confirmability was strengthened through reflexive practices, cross-participant validation, and the IMOI framework. The primary researcher's employment in one case organization remains a potential source of bias.

Regarding *ethical considerations*, participants were informed of the study's purpose, anonymity, and withdrawal rights. Data handling followed established qualitative research guidelines [5].

Finally, our findings are *contextually limited* by a small sample and a single snapshot of hybrid Agile practice. Future research could extend this work through longitudinal or cross-cultural studies or by integrating behavioral or performance data.

7 Conclusion

Hybrid work has transformed how Agile teams collaborate, communicate, and maintain shared understanding. Although it offers flexibility and autonomy, it also introduces risks such as misalignment, uneven participation, and reduced access to informal interactions that are core to Agile processes. Our study shows that hybrid settings reinterpret Agile practices to fit new conditions. We also found out that:

- Hybrid Agile works best when treated as an evolving process.
- It requires new practices rather than direct reuse of co-located routines.
- Effective hybrid work balances synchronous and asynchronous collaboration.
- Cohesion depends on clear communication and equitable participation.
- Psychological safety, trust, and shared ownership remain essential values.

Future work could extend these findings through longitudinal or cross-context studies. Quantitative or behavioral data would complement our qualitative insights by revealing how productivity and collaboration unfold in practice, further clarifying how Agile principles transform in hybrid work environments.

References

1. Adzgauskaite, M., Tam, C., Martins, R.: What helps agile remote teams to be successful in developing software? empirical evidence. *Inf. Softw. Technol.* **177**, 107593 (2025). <https://doi.org/10.1016/j.infsof.2024.107593>
2. Agile manifesto: manifesto for agile software development (2001). <http://agilemanifesto.org/>
3. Behutiye, N.W., Tripathi, N., Isomursu, M.: Adopting scrum in hybrid settings, in a university course project: Reflections and recommendations. *IEEE Access* **12**, 105633–105650 (2024). <https://doi.org/10.1109/ACCESS.2024.3434662>
4. 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.2015.05.008>
5. Flick, U.: *Designing Qualitative Research*. SAGE Publications Ltd (2018). <https://doi.org/10.4135/9781529622737>
6. Hanzis, A., Hallo, L.: The experiences and views of employees on hybrid ways of working. *Administrative Sci.* **14**, 263 (2024). <https://doi.org/10.3390/admsci14100263>
7. Ilgen, D.R., Hollenbeck, J.R., Johnson, M., Jundt, D.: Teams in organizations: from input-process-output models to imoi models. *Annu. Rev. Psychol.* **56**, 517–543 (2005). <https://doi.org/10.1146/annurev.psych.56.091103.070250>
8. Kennedy, I.S., Zadeh, A.A., Choi, J., Alborz, S.: Agile practices and it development team well-being: unveiling the path to successful project delivery. *Eng. Manag. J.*, 1–13 (2024)
9. Ly, D., Overeem, M., Brinkkemper, S., Dalpiaz, F.: The power of words in agile vs. waterfall development: written communication in hybrid software teams. *J. Syst. Softw.* **219**, 112243 (2025)
10. Merriam, S.B., Tisdell, E.J.: *Qualitative research: a guide to design and implementation*, Jossey-Bass: A Wiley Brand (2016). 4 edn
11. 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**(5), 1–41 (2021). <https://doi.org/10.1007/s10664-021-09967-3>
12. Molléri, J.S., Mohagheghi, P.: Transformation to a hybrid workplace: a case from the norwegian public sector. *IEEE Softw.* **41**(5), 70–77 (2024). <https://doi.org/10.1109/MS.2024.3368564>
13. Neumann, M., et al.: What remains from covid-19? agile software development in hybrid work organization: a single case study. In: *2022 10th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pp. 29–38. IEEE (2022)
14. de Souza Santos, R.E., Ralph, P.: A grounded theory of coordination in remote-first and hybrid software teams. In: *Proceedings of the 44th International Conference on Software Engineering*, pp. 25–35 (2022)
15. Steegh, R., Van De Voorde, K., Paauwe, J.: Understanding how agile teams reach effectiveness: a systematic literature review to take stock and look forward. *Hum. Resour. Manag. Rev.* **35**(1), 101056 (2025)
16. Tran, L.: The impact of hybrid work on productivity: understanding the future of work: a case study in agile software development teams. Master's thesis (2022). <https://www.diva-portal.org/smash/record.jsf?pid=diva2:1696127>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Agile Software Engineering Education in the Age of AI: Rethinking Practices, Pedagogies, and Tools



Preparing Students for AI-Driven Agile Development: A Project-Based AI Engineering Curriculum

Andreas Rausch^(✉), Stefan Wittek, Tobias Geger,
and David Inkermann

Clausthal University of Technology, 38678 Clausthal-Zellerfeld, Germany
{andreas.rausch,stefan.wittek,thomas.tobias.marcello.geger,
david.inkermann}@tu-clausthal.de

Abstract. Generative AI and agentic tools are reshaping agile software development, yet many engineering curricula still teach agile methods and AI competencies separately and largely lecture-based. This paper presents a project-based AI Engineering curriculum designed to prepare students for AI-driven agile development by integrating agile practices and AI-enabled engineering throughout the program.

We contribute (1) the curriculum concept and guiding principles, (2) a case study of interdisciplinary, AI-enabled agile student projects, and (3) early evidence from a mixed-methods evaluation. In our case study, second-semester bachelor students work in teams over seven two-week sprints on a realistic software product. AI tools are embedded into everyday agile engineering tasks—requirements clarification, backlog refinement, architectural reasoning, coding support, testing, and documentation—paired with reflection on human responsibility and quality.

Initial results indicate that the integrated approach supports hands-on competence development in AI-assisted engineering. Key observations highlight the need for agile teaching adaptations due to rapid tool evolution, the critical role of oral verification to ensure foundational learning. We close with lessons learned and recommendations for educators designing agile project-based curricula in the age of AI.

Keywords: Agile education · AI engineering · project-based learning · generative AI · case study · evaluation

1 Introduction

Agile ways of working have become a de-facto standard across many engineering domains, increasingly beyond “classical” software projects. At the same time, generative AI and agentic tooling are reshaping how teams implement features, create tests, explore design alternatives, and plan work. As a result, core agile activities—such as backlog refinement, sprint planning, and retrospectives—are

progressively influenced by AI assistance, while questions of quality, responsibility, and trust become more prominent when AI contributes to engineering artifacts.

Despite rising industry desire [1,2], many engineering curricula continue to treat Agile Software Engineering methodologies and AI-enabled development as isolated modules rather than integrated practices. Even where these topics coexist, they are frequently delivered via lecture-centric formats that prioritize theoretical knowledge transfer over situated practice. As of November 2025, data indicates a significant lack of practical application in German higher education: of the 47 AI-centered and engineering-focused programs, only three incorporate project work exceeding 12 European Credit Transfer System (ECTS) points [3]. Notably, none of these three programs recognize Software Engineering as a primary application domain.

This creates a gap: students may learn concepts and terminology, yet rarely experience how AI changes team dynamics, roles, collaboration patterns, and quality assurance in realistic agile projects.

We argue that closing this gap requires more than “AI in a lecture” or “Scrum in theory.” Effective preparation for AI-driven workplaces depends on authentic project settings with evolving requirements, real teamwork with explicit roles and responsibilities, continuous feedback cycles, and deliberate reflection on where human judgment and accountability remain essential. In other words, students need a learning environment in which agility and AI are practiced together as an integrated socio-technical capability.

To address this, we designed and iteratively refined a project-based engineering study program with an explicit AI engineering focus. The program is guided by eight principles: shaping digital transformation; technological innovation with quality and sustainability; human–AI collaboration; project-based learning; agility and reflection; practical relevance; entrepreneurship; and interdisciplinary teamwork. Within the curriculum, students repeatedly conduct AI-enabled agile projects across heterogeneous domains, applying agile practices end-to-end while using AI tools to support engineering tasks (e.g., ideation, coding support, test generation, documentation, stakeholder simulation). At the same time, they explicitly address what must remain a human responsibility, such as acceptance criteria, risk and ethics trade-offs, and accountability.

This paper contributes:

1. A study program concept for AI-enabled, project-based agile engineering education (structure, learning outcomes, guiding principles)
2. A teaching and project concept plus case study describing how AI tools are integrated into agile student projects (roles, events, artifacts, assessment and feedback)
3. An initial quantitative and qualitative evaluation with lessons learned and recommendations for preparing students for AI-driven workplaces

The rest of the paper is structured as follows: Sect. 2 introduces the concept of our agile, project-based AI Engineering study program. Section 3 explains how we conduct interdisciplinary agile projects. Section 4 summarizes and discusses

the initial evaluation results, combining qualitative and quantitative evidence. Finally, Sect. 5 concludes the paper and outlines directions for future work.

2 Designing AI-Integrated Agile Education: AI Engineering Study Program

Teaching agile effectively requires learning formats that go beyond lecture-centric instruction: students need repeated, authentic project settings in which teamwork, evolving requirements, and continuous feedback are experienced rather than explained. At TU Clausthal, we have already operationalized this principle in the project-based study program Digital Technologies, where agile methods (e.g., Scrum), project management, creativity techniques, and teamwork are taught and internalized through practice in real projects [4], [5].

With generative AI, the need for practice-oriented formats becomes even more pressing. LLMs and agentic AI increasingly support—and partly automate—engineering activities, but they also introduce new risks regarding quality, traceability, and responsible decision-making. Current curricula often still prepare students for a “workplace without AI,” creating a qualification gap. We therefore follow the same core educational hypothesis that proved successful for agile education: project-based learning is a powerful vehicle to develop AI competence in context, because it forces students to apply AI tools under real constraints, to reflect on limitations and failure modes, and to take responsibility for outcomes (“human oversight”).

Building on these foundations, we developed the AI Engineering study program concept. Its key idea is to combine a shared core curriculum with discipline-specific specialization tracks. The core curriculum establishes a common baseline across all students—covering programming and software foundations, AI literacy and data-related skills, and agile development practices—so that students can collaborate in interdisciplinary settings. On top of this base, students choose an engineering focus (e.g., AI Software Engineering, AI Mechanical Engineering, AI Material Engineering), in which domain-specific engineering competencies are developed and repeatedly connected to the use of AI as an engineering tool (“AI as a tool for engineering,” not only “AI as a product”).

A distinctive structural element of the program is that a substantial share of the workload is realized in project-oriented formats (approximately one third), where students apply agile methods and AI tools end-to-end on realistic problems—iteratively, in teams, and with continuous feedback. These projects are designed as authentic socio-technical learning settings: students experiment with AI-supported workflows, evaluate results critically, and learn how to embed AI usage into engineering quality assurance and accountability. The curriculum itself is treated as a learning, agile system that is iteratively improved with input from students, educators, industry, and societal stakeholders, and designed for transfer (e.g., reusable materials and modular extensions).

The structure of the study program is summarized in the model curricula shown in Fig. 1: they make explicit (i) the shared core curriculum that establishes

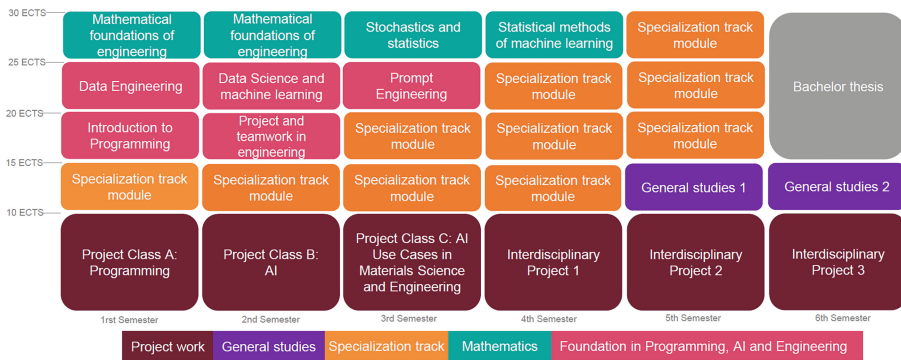


Fig. 1. Model curricula of the bachelor program AI Engineering

a common foundation in programming, mathematics, AI/data competencies, and agile ways of working, and (ii) the subsequent engineering specialization tracks (e.g., AI Software Engineering, AI Mechanical Engineering, AI Material Engineering) that deepen domain-specific knowledge while maintaining a consistent AI-augmented engineering perspective. Importantly, the figures also highlight the program’s strong project-based backbone: project classes and recurring interdisciplinary projects are distributed across the semesters to ensure that students repeatedly apply agile practices and AI tools in authentic team settings—first to build fundamental skills, then to develop and evaluate discipline-specific AI use cases, and finally to integrate them in larger interdisciplinary projects and the bachelor thesis.

3 Case Study: AI-Enabled Agile Projects

The projects we present in this case study are conducted by second-semester bachelor’s students. At this stage, the students possess basic programming skills in Python, knowledge of object-oriented programming in Java, and familiarity with fundamental development tools such as Git. Additionally, the foundations of UML-based modeling, Scrum-based development, and project management have been established in previous courses.

The projects in this case study are conducted by second-semester bachelor students. They enter with basic skills in Python, Java OOP, and Git, alongside foundations in UML modeling, Scrum, and project management established in prior courses.

The core objective is to transform these basics into comprehensive, AI-enabled team development capabilities. Students are expected to derive a domain model and consistent use-case diagrams, formulate epics and user stories, and master estimation techniques. They design a high-level architecture and adopt engineering practices including automated unit testing, clean code, and CI/CD pipelines. Throughout this process, they must demonstrate efficient and critical use of AI tools.

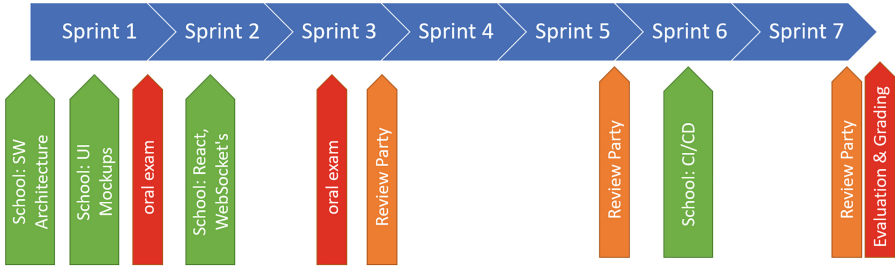


Fig. 2. Time schedule of AI-enabled Agile Projects

Teams consist of 4–6 developers, a Product Owner (teaching assistant), and a Supervisor (professor). While students initially act as developers, they gradually assume Product Owner responsibilities, such as writing user stories. The technical goal is to develop a browser-based multiplayer board game [6]. The project follows the Scrum framework over seven two-week sprints. Standard events include bi-weekly Dailies, mid-sprint “InBetween” presentations, and Sprint Reviews. Additionally, cross-team “Review Parties” are held after sprints 3, 5, and 7 to present results to a wider audience.

Knowledge transfer occurs via “Schools”—hybrid lecture-exercise units focused on technology (e.g., React, TypeScript) and architecture (e.g., client-server concepts). AI assistance is not a separate topic but is embedded into every School. Teachers demonstrate AI capabilities for feedback or code explanation using tools like MS Copilot or ChatGPT, and homework assignments encourage students to use AI tools critically. Figure 2 shows the time schedule of a typical project along with its schools.

To ensure deep understanding despite AI code generation, students undergo individual 20-minute oral exams verifying their code and AI usage. Alongside the project, students maintain a presentation serving as living documentation of requirements, architecture, and process. The final grade is derived from the project repository and this presentation, adjusted by individual peer ratings from a guided final session.

4 Discussion

This project structure was introduced in Summer Semester 2023 within the “Digital Technologies” program [4] (established WS 2019). The WS 2022 cohort was the first to participate. Based on three completed runs, we identified three key lessons:

1. **Agility:** The rapid emergence of production-ready AI tools required agile teaching, requiring to integrate topics like Copilot or agentic frameworks during ongoing projects.
2. **Assessment:** Individual oral examinations are essential to ensure students grasp foundational concepts rather than blindly relying on AI.

Table 1. Average ECTS progression compared to curriculum, as of WS 2025

Intake	Avg. Current ECTS	Expected ECTS	Avg. Diff.	Avg. % Diff.
WS 2021	135	180	−45	−25%
WS 2022	158	180	−22	−12%
WS 2023	115	120	−5	−4%
WS 2024	58	60	−2	−3%

3. **Personification:** Students increasingly personify AI chatbots, occasionally describing ChatGPT as an additional team member.

An initial quantitative evaluation indicates improved student performance starting with the WS 2022 cohort, with the average missing ECTS dropping from 25% (WS 2021) to 3% (WS 2024) (see Table 1).

5 Conclusion

We presented a project-based AI Engineering curriculum that integrates agile practices with generative AI tools to prepare students for modern software development. Our case study demonstrates that authentic, interdisciplinary project settings significantly improve student performance and foster critical AI literacy. As AI tools evolve, future work will focus on adapting assessment models and exploring the long-term impact of AI teammates on agile team dynamics.

References

- Otten, D., Stalnaker, T., Wintersgill, N., Chaparro, O., Poshyvanyk, D., Schmidt, D.: “don’t be afraid, just learn”: Insights from industry practitioners to prepare software engineers in the age of generative AI (2026). [arXiv:2604.06342](https://arxiv.org/abs/2604.06342), [cs.SE]. <https://arxiv.org/abs/2604.06342>
- Chen, A., Huo, T., Nam, Y., Peruma, A., Port, D.: The impact of generative AI-powered code generation tools on software engineer hiring: recruiters’ experiences, perceptions, and strategies. GSCC: 0000006 2026-04-15T14:29:32.308Z 0.09 (2025). <https://doi.org/10.24251/HICSS.2025.864>, <https://hdl.handle.net/10125/109716>. Accessed 15 Apr 2026
- ZEIT ONLINE: Zeit campus studiengänge - orientierung für studium und beruf. <https://studiengaenge.zeit.de/>. Accessed 15 Apr 2026
- Center for Digital Technologies: Digital Technologies. <https://digitecstudieren.de>. Accessed 20 Jan 2026
- Stenzel, F., Küpper, S., Rausch, A., Schiering, I., Bikker, G.: Projektbasierte lehre in der informatik zur entwicklung von zukunftscompetenzen,” de, in *Tagungsband zum 6. Mint Symposium: Zukunft MINT Lehre: Was bleibt? Was kommt? Was wirkt?*, H. Dölling and C. Schäfle, Eds., München: BayZiel, pp. 170–177 (2025). https://doi.org/10.57825/repo_in-6376
- Center for Digital Technologies: Zweitsemesterprojekte. <https://projects.digitecstudieren.de>. Accessed 20 Jan 2026

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Teaching Agile Requirements Engineering: A Stakeholder Simulation with Generative AI

Eva-Maria Schön¹✉, Michael Neumann², and Tiago Silva da Silva³

¹ University of Applied Sciences Emden/Leer, Emden/Leer, Germany
eva-maria.schoen@hs-emden-leer.de

² University of Applied Sciences and Arts Hannover, Hannover, Germany
michael.neumann@hs-hannover.de

³ Federal University of São Paulo, São Paulo, Brazil
silva.tiago@unifesp.br

Abstract. *Context:* The active involvement of users and customers in agile software development remains a persistent challenge in practice. For this reason, it is important that students in higher education become familiar with good practices in Agile Requirements Engineering during their studies. *Objective:* Our objective is to support students in learning how to perform requirements analysis by applying agile practices within a stakeholder simulation using AI Personas. At the same time, the approach fosters a critical understanding of the limitations and ethical considerations associated with the use of Generative Artificial Intelligence (GenAI) in this context. *Method:* In our courses, we employ a stakeholder simulation using GenAI, in which students conduct interviews with AI Personas through a provided meta-prompt. Based on the outcomes of these interviews, students apply agile practices (e.g., story mapping or impact mapping) to document requirements. The use of GenAI is subsequently reflected upon in a structured group discussion. *Results:* Through this approach, students gain practical experience by applying state-of-the-art agile practices for requirements elicitation and documentation while simultaneously developing an understanding of the technical and ethical limitations associated with the use of generative AI. *Conclusion:* We have applied this approach over several terms and found that using a meta-prompt provides flexibility, allowing us to remain independent of specific large language model providers.

Keywords: AI Personas · Agile Requirements Engineering · generative AI · education

1 Introduction

The disruptive changes triggered by the public availability of Generative Artificial Intelligence (GenAI) at the end of 2022, following the release of ChatGPT [10], have not yet reached their peak. In recent years, experimentation

with this technology has strongly shaped higher education, with lecturers and students jointly gaining hands-on experience and learning which applications are effective and where limitations emerge. GenAI has fundamentally altered how teaching and learning take place, with particular implications for higher education in the field of software engineering [9]. One insight has become increasingly clear: without sufficient knowledge of AI literacy and ethical considerations, this powerful technology is often misused, since existing guidelines are often ignored [2].

Motivated by this observation, our goal is to integrate GenAI into the classroom in a meaningful and pedagogically grounded manner. To this end, we have been using GenAI for teaching Agile Requirements Engineering (RE) for several terms (since winter term 24/25, Nov. 2024). Agile RE follows an iterative approach across the entire product development lifecycle [14]. Requirements are treated as hypotheses that are continuously refined through ongoing feedback from stakeholders and users [13]. Despite this iterative nature, practitioners encounter various challenges when performing RE activities in real-world settings [4]. Recent research further indicates that the active involvement of users and customers remains a persistent challenge in practice [7]. However, we use stakeholder simulations based on *AI Personas* to demonstrate the iterative nature of Agile RE and to address the advantages and disadvantages of using GenAI in this context.

The persona method has long been established in Agile RE [13] and Human-Centered Design [5] to address the needs of different target groups during the development of interactive systems. *AI Personas* (also known as “synthetic users” [12]) extend this approach by implementing personas through GenAI chatbots such as ChatGPT. These AI Personas are synthetic constructs designed to emulate human-like characteristics and behaviors tailored to specific roles within GenAI systems [6]. However, the use of AI Personas also entails limitations. Personas are fictional characters that can represent users in the product development process. However, they do not replace interactions with real users. This also holds for AI Personas. Since the data underlying these AI Personas originates from the training data of large language models (LLMs), they may reflect inherent biases. Consequently, the use of AI Personas for the actual development of interactive systems remains limited, making their application particularly suitable for educational and reflective contexts.

This paper is structured as follows: First, we provide an overview of our teaching case and offer teaching tips for other lecturers on how to apply it in Sect. 2. Section 3 reflects on our experiences with its use and share our learnings related to didactic design and technical implementation. At the end, we close the paper with a conclusion and outline our future work in Sect. 4.

2 Teaching Case and Teaching Tips

2.1 Learning Objectives

In our courses, we often formulate learning objectives as user stories [3] to clearly express the relevance of the tasks from the students’ perspective. The acceptance

criteria are described in the form of learning outcomes aligned with the revised learning objectives taxonomy of [1]. Figure 1 presents the learning objectives for the stakeholder simulation using generative AI. This user story is discussed with the students at the beginning of the exercise to ensure that the learning objectives of the activity are clearly understood.

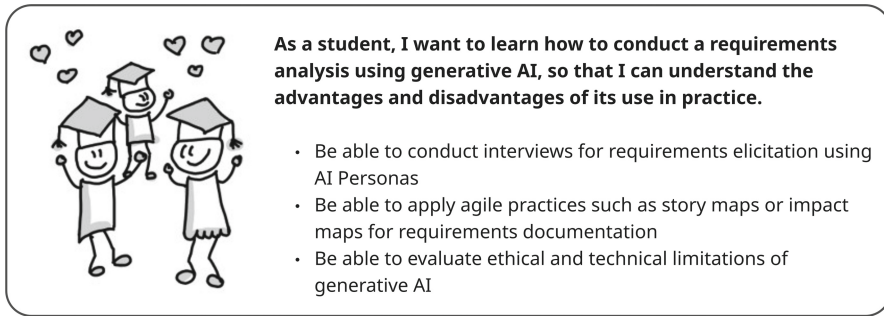


Fig. 1. Learning objectives formulated in form of a user story.

2.2 Teaching Tips for Application of the Teaching Case

This section outlines teaching tips for the application of the teaching case. The teaching case can be applied to both on-campus and online study programs. The stakeholder simulation consists of three elements that are described in the following (see Fig. 2).

1. Requirements Discovery. In the first step, students receive the task description. This includes a case study (e.g., the development of a mobile application for the digitalization of citizen-oriented public services). In addition, students are provided with a meta-prompt that can be used with any GenAI chatbot, such as ChatGPT. Once the meta-prompt is entered into the chatbot, it simulates the preconfigured AI Personas representing the relevant stakeholders. Students can then conduct interviews with these stakeholders and ask questions (e.g., *What problems do citizens experience with the current digital service offerings?*). The stakeholders respond according to their predefined roles configured in the meta-prompt.

2. Requirements Specification. In the next step, students are asked to transfer the requirements collected during the interviews into appropriate artifacts. Various artifacts from Agile RE can be used for this purpose. Based on our experience, visual artifacts such as story maps and impact maps are well-suited, as they cannot be easily generated by applying GenAI tools. When students are asked to create user stories, it should be specified that these user stories are to be refined through collaborative discussion.

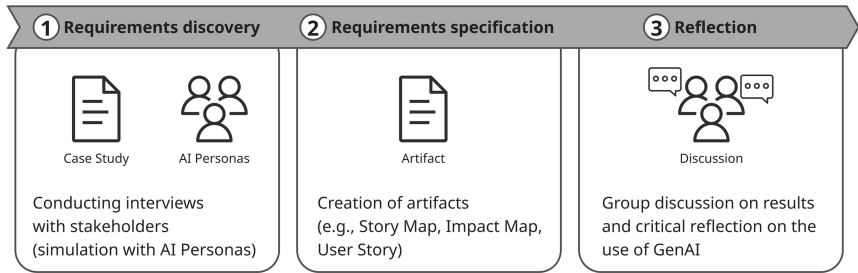


Fig. 2. Overview of the elements of the teaching case.

3. Reflection. The third step involves critically reflecting on the learning process to strengthen the transfer of theory and practice. During a group discussion in which all teams are brought together, students reflect on their experiences with the stakeholder simulation, including the limitations of GenAI, and discuss ethical considerations related to the use of GenAI. Suitable discussion topics include:

- Discussion of the quality of the artifacts produced for requirements specification and the application of agile practices (*e.g.*, when applying user stories, the team discussion about the requirement is important for developing a shared understanding, and not only the production of the written part of the user story).
- The use of AI Personas for Agile RE has some limitations because of the bias in training data of the LLMs, which may result in stakeholder representations that do not reflect real target groups, leading to requirements that do not align with the needs of actual users. Though this aspect should be considered when applying AI Personas, we are aware of the limitation of a persona's mismatching to target groups when using the personas concept in general.
- Limitations of GenAI, such as losing contextual information, responding in a way that prioritizes efficiency over human-like communication, and the occurrence of hallucinations.
- Biased training data for LLMs (*e.g.*, over-representation on one gender or under-representation of specific groups such as neurodivergent persons) should be covered.
- Depending on the module and the country in which the teaching case is embedded, it may also be appropriate to include a discussion on data privacy and regulations such as the EU AI Act and GDPR.

The teaching tips outlined above refer to the example of embedding the teaching case presented in Sect. 2.3. The modular structure of the teaching case allows it to be easily adapted to different contexts, such as another module, a different case study with a different target group, or the use of alternative agile practices.

2.3 Example of Embedding the Teaching Case

In this Section, we describe our teaching case on stakeholder simulation using GenAI, which we have already applied in several modules. The following section illustrates a concrete implementation of the case within a specific module. The here presented teaching case was applied during the 2024/2025 winter term at the University of Applied Sciences and Arts Hannover. An overview of the instructional setting and implementation conditions is presented in Table 1.

Table 1. Overview of the Teaching Case

Topic	Description
Degree programs	Bachelor of Science (B.Sc.): – Business Information Systems (BIS) – E-Government (EGOV)
Module	Requirements Analysis (3rd term of studies)
Learning Objectives of Module	The students are able to analyze requirements for software products by applying the object oriented paradigm using Unified Modeling Language.
Students	The group size is around 65 students for BIS and 30 students for EGOV. The students have diverse backgrounds in terms of profession and school examination.
Implementation of Teaching Case	The teaching case is implemented as an impulse workshop. Attendance at the workshop is not mandatory.
Duration of case	The workshop is planned for 90 minutes.
Term of application	Winter term 2024/2025

As part of a teaching case implemented in the winter term 2024/2025 at the University of Applied Sciences and Arts Hannover, students were trained to elicit and analyze requirements for a mobile public administration app. The objective was to prepare prospective requirements engineers to work with different user types and to understand the challenges of digital transformation in the public sector. The scenario focused on typical citizen services, such as applying for identity documents or registering a change of residence, while reflecting the scepticism and change-resistance often found in public administrations. At the same time, the involved stakeholders were intrinsically motivated to support the requirements engineer and provide realistic requirements.

Students conducted elicitation sessions with three predefined stakeholders representing different user perspectives: A public administration specialist, an accessibility-focused citizen, and a power user from a large city. Each stakeholder provided concise requirements for an appointment booking feature, expressed preferences, and occasionally showed resistance by posing counter questions or referencing failed app functions from well-known administrations. The students were expected to respond empathetically, stay within the defined project context, and steer the conversation back to the elicitation goal when needed.

Students applied elicitation methods (e.g., interviews), analyzed the resulting requirements, and documented them in user story format in a product backlog. The exercise enabled them to practice realistic stakeholder communication, handle conflicting expectations, and translate elicited information into structured agile requirements.

3 Learnings and Experiences of the Practical Application

We have already applied the teaching case in various on-campus and online study programs and experimented with different configurations. This paper reflects our observations and experiences gathered during its implementations in different scenarios. On the one hand, we summarize our experiences related to the didactic design and the use of agile practices; on the other hand, we reflect on our experiences with the technical implementation of AI personas using generative AI tools.

3.1 Learnings on Didactic Design and Use of Agile Practices

We have applied the teaching case in different modules and across various study programs. In this context, we gained experience with the use of different agile practices for element 2. Requirements specification (see Fig. 2).

User Stories [3]. After completing the interviews, students used GenAI to generate user stories. In this step, additional advice from the lecturer was necessary to encourage students to discuss and refine the generated user stories within their groups. In several groups, it became apparent that the need for refinement was not initially recognized, as the GenAI-generated outputs sounded plausible and well-formulated. However, students did not realize that the user stories were too superficial for implementation in software development and lacked measurable acceptance criteria, e.g., documented as Definition of Ready.

Story Maps [11]. When students created story maps manually, the results were in line with the requirements of a story map. In contrast, when GenAI was used to generate story maps, two challenges became apparent. First, advanced prompting skills were required to obtain a visual story map template as output. Second, a story map is intended to tell a coherent story from the user's perspective. With novice prompting skills, this proved difficult to achieve, as the GenAI-generated story maps tended to resemble collections of features grouped by category rather than user-centered narratives.

Impact Maps [8]. When creating the impact map, some students experienced difficulties distinguishing between the concept of stakeholders used in the exercise and the concept of actors as defined by the agile practice of impact mapping. Contrary to the assumptions of some students, there is no one-to-one relationship between these concepts. In addition, the results produced by the student groups indicated challenges in identifying an appropriate level of granularity for the deliverables. Some groups listed non-functional requirements such as accessibility

and user experience, while others focused on concrete features, such as user profiles or dashboards. When the exercise was conducted in September 2025, the LLMs in use were already capable of generating impact maps that met the requirements of the exercise based on the interview data, even with basic prompting skills. In this context, we recommend creating this artifact in an in-class setting, as it can otherwise be generated using GenAI without sufficient individual effort when assigned as homework.

Overall, it is advisable to revisit and critically reflect on the experiences students gain during the requirements specification phase in element 3. Reflection (see Fig. 2). Otherwise, there is a risk that students may apply agile practices incorrectly and produce low-quality results.

Additional feedback from students indicated that they initially struggled to understand their role within the task. We therefore recommend explicitly emphasizing, when introducing the assignment, that students assume the role of a requirements engineer who conducts interviews with stakeholders and is responsible for translating the elicited requirements into a specification.

3.2 Learnings on Technical Implementation

The LLMs provided by different GenAI providers are evolving rapidly. As a result, there are multiple ways to make stakeholder simulations using AI Personas available to students. In March 2025, we initially started with a customized-GPT hosted through OpenAI. Students were able to access the simulation via a direct link to the OpenAI platform. However, this approach led to several limitations related to licensing conditions. Students using a basic OpenAI account were only able to send a limited number of prompts to the chatbot (approximately five) and then had to wait several hours before continuing the interaction. This restriction significantly impaired the learning experience. In addition, some students expressed concerns regarding data protection and indicated a preference for using local GenAI models or alternative providers. As a result, we transitioned to providing the stakeholder simulation through a meta-prompt. This approach allows students to decide independently which GenAI tool they wish to use to run the simulation. So far, we have had positive experiences with this approach.

To adapt the meta-prompt to newer LLMs, we optimized it in September 2025 using the Prompt Optimizer provided by OpenAI. This optimization noticeably changed the behavior of the AI Personas. Communication became strongly efficiency-oriented and less interactive. The AI Personas provided the requested information in a highly targeted manner and asked very few follow-up questions. Consequently, the AI Personas were perceived as less human-like, and the students finished their interviews quickly.

At present, we are experimenting with different versions of the meta-prompt to make the behavior of the AI Personas more human-like and to assign them distinct personalities.

4 Conclusion and Future Work

This paper presents our learnings and experiences from a teaching case that integrates stakeholder simulation with AI Personas into Agile Requirements Engineering education in higher education. By combining interviews with stakeholders that are modeled as AI Personas with agile practices for requirements discovery and specification, our approach enables students to gain hands-on experience while critically reflecting on the technical and ethical limitations of generative AI. Our experiences from multiple course implementations indicate that AI Personas can serve as a valuable pedagogical instrument when used in a structured and reflective manner.

Future work will focus on further refining the meta-prompts used to control the behavior and personality of the AI Personas in order to support more natural and interactive stakeholder simulations. In this context, we are also evaluating the use of agentic AI. We are also experimenting with different forms of interaction, such as interviews via ChatGPT's voice interface. In addition, we plan to systematically evaluate learning outcomes across different course formats and student cohorts, as well as to explore extensions of the teaching case to other software engineering and project-based learning contexts.

Acknowledgements & Final Remark. The teaching material can be provided upon request. Finally, we note that text-parts of this paper were refined with the support of the GenAI tools ChatGPT (GPT-5.2) and DeepL.

References

1. Anderson, L., Krathwohl, D.: A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives: Complete Edition, Longman (2001)
2. Bischof, L., Schön, E.M., Rauschenberger, M., Neumann, M.: "we need to analyze students genAI use": Towards an AI adoption framework for higher education. In: Proceedings of the 21st International Conference on Web Information Systems and Technologies, pp. 429–438 (2025). <https://doi.org/10.5220/0013819000003985>
3. Cohn, M.: User Stories Applied: For Agile Software Development (2004). <https://books.google.com/books?hl=de&lr=&id=SvIwuX4SVigC&pgis=1>
4. Fernández, D.M., et al.: Naming the pain in requirements engineering. *Empir. Softw. Eng.* **22**(5), 2298–2338 (2016). <https://doi.org/10.1007/s10664-016-9451-7>
5. International Organization for Standardization: ISO 9241-210:2019(en) ergonomics of human-system interaction—part 210: human-centred design for interactive systems (2019)
6. Karimova, G.Z.: Humanizing AI with Personality. Springer, Cham (2025). <https://doi.org/10.1007/978-3-031-82327-5>
7. Kollmorgen, J., Pilz, J., da Silva, T.S., Thomaschewski, J., Schön, E.M.: State of the art of agile user experience: challenges and solution approaches. *Comput. Stand. Interfaces* **96**, 104083 (2026). <https://doi.org/10.1016/j.csi.2025.104083>
8. LLP, N.C.: Impact mapping (2026). <https://www.impactmapping.org/>

9. Neumann, M., Rauschenberger, M., Schön, E.M.: “we need to talk about ChatGPT”: the future of AI and higher education. In: 5th International Workshop on Software Engineering Education for the Next Generation (SEENG), pp. 29–32 (2023). <https://doi.org/10.1109/SEENG59157.2023.00010>
10. OpenAI: ChatGPT: optimizing language models for dialogue (2022). <https://openai.com/blog/chatgpt/>
11. Patton, J.: User Story Mapping: Discover the Whole Story, Build the Right Product, First Edit edn. O’Reilly (2014)
12. Pehar, F.: AI as synthetic users in human-centred design: a systematic review. In: 2025 MIPRO 48th ICT and Electronics Convention, MIPRO 2025 - Proceedings. pp. 1500–1509. Institute of Electrical and Electronics Engineers Inc. (2025). <https://doi.org/10.1109/MIPRO65660.2025.11131969>
13. Schön, E.M., Thomaschewski, J., Escalona, M.J.: Agile requirements engineering: a systematic literature review. Comput. Stand. Interfaces **49**, 79–91 (1 2017). <https://doi.org/10.1016/j.csi.2016.08.011>
14. Sporse, T., Dingsøyr, T., Stol, K.J.: User stories as boundary objects in agile requirements engineering: a theoretical literature review. J. Syst. Softw., 112693 (2025). <https://doi.org/10.1016/j.jss.2025.112693>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter’s Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter’s Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Introducing Agile Team Practices in Capstone Courses: Impact on Student Collaboration

Viktoria Stray^(✉) , Yngve Lindsjörn , Idunn Stabell ,
Camilla Kielland Lund , Steffen Almås , and Marthe Berntzen 

University of Oslo, Oslo 0373, Norway

{stray,ynglin,idunnsta,camilltk,steffa,marthenb}@ifi.uio.no

Abstract. Software engineering capstone courses provide students with real-world team project experience, but active and even collaboration within teams remains hard to achieve. In this experience report, we describe how incorporating activities into the assessment structure of a large undergraduate capstone course changed students' teamwork experiences. In 2024, agile practices, such as pair programming, daily stand-up meetings, and collaborative planning sessions, were integrated into the grading system, partly to address collaboration issues and partly to respond to increased use of Generative AI tools like ChatGPT. We analyzed data from 872 student surveys across four course offerings from 2022 to 2025, together with 12 semi-structured interviews. Across surveys and interviews, students reported that the agile practices were associated with greater accountability, reflection, and participation, and made collaboration more visible and harder to postpone. In 2025, 74% of students described AI strictly as a tool rather than a team member, with AI use concentrated in debugging, concept explanation, and design discussion. We discuss implications for collaboration-focused assessment in capstone courses in the context of growing AI use, and provide an overview of the team activities as an operational template for educators.

Keywords: SE Education · Collaboration · Team dynamics · ScrumBan · Generative AI · Stand-up meetings · Retrospective · Assessment strategies

1 Introduction

Teamwork is central to both Software Engineering (SE) education and professional practice. This is particularly visible in agile development, where collaboration, iterative delivery, and self organization are core expectations. In SE education, capstone project courses are widely used to help students connect academic concepts to the practical realities of developing software in teams [2, 9]. Beyond technical learning, these courses are meant to develop interpersonal capabilities that are difficult to teach in isolation, including communication, coordination, conflict management, and leadership [7].

© The Author(s) 2027

A. Goldman et al. (Eds.): XP 2026 Workshops, LNBIP 596, pp. 171–180, 2027.

https://doi.org/10.1007/978-3-032-39205-3_18

The spread of AI-assisted tools adds a new variable to team collaboration and learning. The growing use of generative AI (GenAI) tools like ChatGPT and GitHub Copilot is already changing how agile practitioners work [4]. These tools can raise individual productivity, but the implications for teamwork and learning are mixed. Studies report that GenAI tools are reshaping how developers work, both solo and in pairs, with developers often turning to AI for individual tasks while still preferring peers for complex problem solving [11].

Further, GenAI use is affecting the learning environment in higher education [14]. For example, written reports, which used to be a central assessment artifact in capstone courses, are now easier to produce with AI assistance and can provide weaker signals about students' learning. This can also increase the burden on instructors, since many reports become more similar and less exciting to read. Educators may need to shift emphasis away from deliverables that are easily automated and toward human-centred skills such as critical reflection, collaboration and judgement [3].

Agile approaches such as Scrum and Kanban are widely used in educational settings because they provide a concrete collaboration structure without removing student autonomy. For instance, studies show that Scrum is frequently adapted in practice, and that teams vary how roles and practices are enacted across contexts [5].

One increasingly used approach to incorporate collaboration practices in educational SE project work is that instructors intentionally embed and support specific agile practices, such as daily stand-ups, sprint planning, and retrospectives, into the coursework [8]. Industry research finds that even in mature agile organizations, practices such as pair programming do not become habits on their own; they depend on first-hand positive experiences, supportive routines, and a team culture that values the practice [1]. The use of agile practices can create regular opportunities to surface issues and support the student teams in self-regulation. In addition, by shifting emphasis from final artifacts (such as reports) toward observable team processes, they can help maintain authentic learning outcomes in settings where AI support is readily available.

This study investigates how incorporating agile practices is perceived by students and teaching assistants (TAs) in an SE capstone course context. Building on a longitudinal dataset from four course offerings (2022–2025), the study aims to present our experiences with introducing agile team practices in such courses, and provide actionable guidance for SE educators.

2 Methodology

The course "Software Engineering with Project Work" is a 20 ECTS capstone undergraduate course at the University of Oslo at the Department of Informatics. The course has grown rapidly since its 2018 pilot, and in 2025, we had 343 students across 57 project teams. The primary course goal is for students to move beyond theory to experience the full software development lifecycle while utilizing agile methods. For most students, this is the first time they participate

in a large software project, so the course emphasizes learning to work effectively as a team. The course consists of two phases. First, students complete an 8-week individual module focused on programming, architecture, version control, APIs, agile theory, and teamwork. This is followed by a 10-week team project in which student teams develop a comprehensive app and produce a project report. The team project counts for 50% of the final course grade; the other 50% is based on an individual written exam without any aids which assesses individual learning outcomes.

Before 2024, the project grade was based on the final report (70%) and the application (30%), with no graded team activities. This structure created a documentation bottleneck where one or two students often carried most of the writing. Another concern was that reports increasingly reflected AI-assisted writing rather than student learning. From 2024, the grading scheme was changed to shift weight from the report toward team activities: the report and the application each account for 35%, while team activities account for the remaining 30%. Agile activities such as retrospectives, pair and mob programming, stand-ups, sprint review, and sprint planning were integrated into the grading criteria. Activities were documented in a process log and discussed with the team's assigned TA. The TAs reviewed each log entry and, in weekly team meetings, asked the team to describe how the activity had been carried out and how it influenced subsequent work. Points were available only when an activity was both documented in the log and discussed with the TA, with the final award made by the course examiners after submission. Some activities were mandatory, while others were optional, to give teams flexibility to choose what best suited their needs.

To evaluate this change, we combined quantitative and qualitative data from 2022 to 2025. We collected 872 survey responses across four end-of-course surveys (2022, 2023, 2024, 2025). The surveys included items on hours spent on the project, agile method used, satisfaction with team activities on a five-point Likert scale, and use of AI tools. Core items were kept consistent across years; items on AI use were added from 2023 onward. The survey data was supplemented by 12 semi-structured interviews with students and TAs in 2024 and 2025. Interviews were transcribed and thematically analysed. The interview guide is accessible online at: <https://doi.org/10.5281/zenodo.15025427>. While prior analyses of the same course addressed aspects such as pre-existing relationships [10], and gender diversity [12], we focus here on how the agile team practices introduced in 2024 shaped collaboration.

3 Results

3.1 Agile Method Use and Adoption of Mandatory Activities

The majority of students across all years reported following ScrumBan (see Table 1). In 2022 and 2023, using agile practices was voluntary, which did not work satisfactorily for all teams. Some struggled with knowing how to apply agile methods without more concrete guidance from the course. One 2022 student described: *"Honestly, we had no clear list or structure. We knew retros and*

Table 1. Use of agile methods

Agile Method	2022	2023	2024	2025
Scrumban	81.1%	83.2%	65.8%	85.4%
Scrum	12.4%	7.4%	24.3%	11.0%
Kanban	6.5%	7.4%	6.3%	3.7%
Other	0.0%	2.0%	3.5%	0.0%

stand-ups existed, but beyond that, it was a lot of guessing and googling to figure out what agile practices even were. That took a lot of time we didn't really have" [I12].

Partly because of this, in 2024, we introduced activities as part of the course-work, including completing selected agile practices (see Table 2). The activity component contributes 30% of the project grade, with a maximum of 30 points; teams choose which optional activities to complete on top of the seven mandatory ones. We selected activities to cover different aspects of agile teamwork: team start-up, planning, development, reflection, and quality assurance. Mandatory activities (mob programming, retrospective 1 and 2, MVP demo, user insight, user testing, GenAI improvement activity) were those that required the whole team to function together. Optional activities (such as TDD, guerrilla testing, and backlog refinement) added value for some projects more than others. To encourage steady work across the project, teams could earn points for at most three activities per week. Teams could reach 30 points without completing every optional activity. Activities were documented in a process log that included rationale, participants, outcomes, and reflection, and was assessed after submission rather than during the project.

The students reported that the required activities helped their teams collaborate more consistently. One student emphasized that although the activities were "a bit stressful," they were worthwhile because they led the team to experiment with mob programming and pair programming, which they likely would not have adopted without the assignments and ended up appreciating. Another student stated, *"The team activities made us meet more often, simple as that. It wasn't just about the points; we got work done. And writing logs early helped when we later had to explain our process in the report"* [I3].

Daily Stand-Up Meetings. Across all years, those who conducted daily stand-up meetings were satisfied with the practice (see Table 3). Several students described daily stand-ups as supporting transparency, shared awareness, and a more even distribution of contribution. A female student from 2023 noted how stand-ups helped create a sense of shared understanding within her team: *"Stand-up was really useful. It made sure everyone knew the plan and what the others were working on. It also made it easier to ask each other for help"* [I5].

Table 2. Team activities with required status and point allocation. The table provides an operational template for educators adopting a similar scheme.

Activity	Purpose and expected outcome	Req.	Pts
<i>Throughout the project</i>			
Weekly supervisor meetings	Maintain regular supervisor contact and document action points after each meeting; points available for up to five weeks.	No	5
Agile meetings	Conduct two different agile meetings, such as standup, sprint planning, or sprint review, in different weeks.	No	2
Self-defined agile activity	Carry out and document a self-defined agile activity not otherwise included in the scheme.	No	1
<i>First half of project (Weeks 10 to 14)</i>			
Kick off activities	Establish shared norms and understanding through team activities (Stinky Fish, Barometer).	No	1
Mob programming	Whole team development to practise collective ownership and coordination (min. 1 h).	Yes	2
User insight	Collect and synthesize user input (e.g., interviews, surveys).	Yes	2
Team workshop	Structured collaboration (for example, Crazy 8s) to move the project forward in the current phase.	No	2
Retrospective 1	Reflect on collaboration and agree on concrete improvement actions.	Yes	2
MVP or PoC demo	Demonstrate a working Minimum Viable Product (MVP) or Proof of Concept (PoC) and align on next steps.	Yes	3
Architecture sketch and data flow report	Communicate MVP structure, responsibilities, and data flow.	No	1
<i>Second half of project (Weeks 15 to 19)</i>			
Retrospective 2	Revisit issues and actions since Retrospective 1, and decide next steps.	Yes	2
GenAI improvement activity	Use GenAI for improvement (e.g., refactoring) and reflect on how it contributed.	Yes	2
Backlog enrichment	Make quality concerns explicit using quality tags to support prioritisation.	No	2
Pair programming session	Practice close collaboration and knowledge sharing while implementing.	No	1
User testing	Evaluate functionality and usability, produce a prioritised change list.	Yes	2
Guerrilla testing	Rapid feedback from an external participant, update backlog accordingly.	No	2
Test driven development (TDD)	Apply TDD to ViewModel, Repository, and API related features.	No	2
Backlog refinement	Ensure a prioritised and estimated backlog for upcoming sprints.	No	1
Req. = mandatory team activity; Pts. = number of available points.			

A male student from 2024 similarly emphasized how regular meetings kept the team aligned and motivated: *“Daily stand-ups made a real difference for our team. They kept us on the same page, maintained momentum, and pushed us to get more done. Even if every stand-up wasn’t perfect, they kept us focused and moving forward together”* [I3].

3.2 Course Effort

As shown in Table 3, the survey responses indicate that students in 2024 and 2025 spent more time on the project than in 2022 and 2023. The TAs reported lower absence rates in the same period. We argue that more regular meetings and collaborative practices likely contributed to this pattern, though cohort-level effects (such as increased AI use) may also play a role. Another common benefit was the motivational effect of tying the activities to grading. Students reported being more engaged and consistent in their efforts: *“In other courses,*

Table 3. Hours spent and satisfaction with stand-up meetings.

Metric	2022–2023	2024	2025
Weekly hours spent on the agile project	19.83	22.6	21.5
Satisfaction with stand-up meetings	4.2	4.4	4.2

Note. Satisfaction was measured on a five-point Likert scale, from 1 (not satisfied) to 5 (very satisfied).

where these kinds of things are optional or not essential for assessment, well, if you do them, you tend to do them lazily. And in many cases, people just write that they did something even if they didn't. So I doubt we would have done as much of it if it hadn't been part of the assessment. I'm really happy with that" [I11].

However, the benefits of these activities were not equally distributed. Several TAs observed that the structured activities had the largest effect on teams that had previously struggled to establish their own routines.

3.3 Use of GenAI Tools

In the survey, students were asked, "If the team used AI in the project, was it viewed as a team member or as a tool?" In total, 74.4% of respondents described AI used only as a tool, while 18.3% stated to use it both as a tool and considering it to be a team member. Only 7.3% reported not using AI in their project. Table 4 provides an overview of what the 2025 students reported using AI tools for in their projects.

In the interviews, some students explained that their teams used AI tools during the early coding phases, particularly to help set up basic structures or understand new coding languages. However, as the projects progressed and the technical complexity increased, the usefulness of AI often declined. One student stated: *"We used AI quite a bit. Toward the end, the code got too complex for ChatGPT to be very helpful. But in the beginning, it was really useful for setting up basic structures or figuring out unfamiliar code. We didn't use it for the report, but we did use it partially during coding and on some assignments"* [I10].

4 Discussion

Across four cohorts, the introduction of mandatory agile team practices was associated with stronger reported student satisfaction and more visible team collaboration. Generally, team members found these activities helpful. Stand-ups were particularly mentioned in the interviews, and participants noted that they valued them because they got insight into everyone's contributions and promoted a sense of responsibility. It was noted that the clear agenda of these meetings influenced the teams positively, by facilitating a better overview which led to equal participation. This is in line with studies that emphasize how structured

Table 4. Reported areas of AI use in student teams in 2025.

Area of AI use	Example statements from student survey
Debugging and error handling	"AI helped us understand error messages in Logcat", "When errors occurred, we used AI to investigate them", "If there was a bug we could not find, AI often spotted it quickly"
Code support in small, isolated parts	"Used mostly for smaller functions rather than entire files", "Help with syntax, finding errors, and a bit of design"
Concept explanation and learning support	"Used to explain concepts and the reasoning behind certain choices", "Helped us understand Kotlin and difficult concepts."
Libraries and poor documentation	"Used when we needed guidance on libraries with poor documentation", "AI helped explain Jetpack Compose elements that were hard to look up"
Design and solution discussion	"AI was used for brainstorming related to code and design", "A discussion partner when thinking about solutions"
Limited code generation	"Sometimes it generated very simple components, like buttons", "We were critical of the generated code examples"
Documentation and text support	"Used to comment code and generate documentation", "Rewriting sentences and fixing language in the report"
Process and team activities	"Templates for interviews, retrospectives, and similar activities", "Only used to complete the mandatory AI team activity"

communication helps foster inclusive teamwork [6]. A similar pattern extended to mob and pair programming. Several students reported trying these practices only because they counted toward the grade, and ended up appreciating them. This mirrors findings from industrial agile teams. Andersen et al. [1] studied how a software company integrated pair programming as a work practice and found that first-hand positive experiences are essential for sustained engagement, while the expectations formed by these experiences shape whether developers continue to pair. We argue that graded team activities can serve a comparable function in student teams to what supervised trial periods serve in industry: they create the first-hand experiences a collaborative practice needs to take root.

4.1 Recommendations for Educators

Our experience suggests that SE capstone courses can support collaboration by making agile teamwork practices explicit, assessed, and discussed throughout the project. Such an approach also helps students internalize good practices for their future careers [2, 7, 9]. This also helps shift assessment toward human-

centred skills and interactions that are harder to automate with GenAI [3,14]. We offer four practical recommendations:

- **Embed agile teamwork activities into the course:** Use structured agile team activities (see our suggestion in Table 2) to foster a culture of psychological safety, feedback, and shared responsibility in student teams. Make these activities explicit parts of the course (for example, via participation points or deliverables).
- **Align assessment with collaboration process:** Ensure that grading criteria reward observable collaboration processes and not only final artifacts such as code and reports. Provide explicit guidance on teamwork expectations and common collaboration challenges early in the course, so students know what is expected and can address issues before they escalate.
- **Encourage continuous reflection:** Incorporate periodic teamwork reflection through e.g., retrospectives, peer feedback surveys, or journals. Structured reflection prompts about effort distribution, role dynamics, and team well-being can help students identify and address problems early. Assign TAs in a non-grading coaching role, with weekly team meetings where the TA helps the team reflect on progress, surface challenges, and decide on action points. This role mirrors how agile coaches in industry support teams in translating reflection into concrete improvements in ways of working [13].
- **Guide responsible use of AI in teamwork:** Rather than banning AI tools, teach students when and how to use them in a way that supports the team. Provide guidelines that AI should complement teamwork (e.g., as a brainstorming tool or for creating boilerplate code). Encourage teams to discuss AI output together, also when pair and mob programming, to maintain focus on learning and openness around the use of AI tools.

5 Conclusion

This study shows that educators can create a learning environment that mirrors the collaborative nature of professional software development while also nurturing the skills and mindsets needed for students to thrive in an era of new technologies. The pattern we observed across the four cohorts gives us confidence that activity-based assessment can support teamwork in capstone courses. We invite other educators to adapt the scheme in Table 2 to their own settings, and to report what works and what does not.

References

1. Andersen, N.H., et al.: Integrating pair programming as a work practice. *J. Syst. Softw.* **234**, 112730 (2026). <https://doi.org/10.1016/j.jss.2025.112730>
2. Bastarrica, M.C., Perovich, D., Samary, M.M.: What can students get from a software engineering capstone course? In: 2017 IEEE/ACM 39th ICSE: Software Education and Training Track, pp. 137–145. IEEE (2017)

3. Daun, M., Brings, J.: How ChatGPT will change software engineering education. In: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education vol. 1 (ITiCSE 2023), pp. 110–116. ACM (2023). <https://doi.org/10.1145/3587102.3588815>
4. Diebold, P.: From backlogs to bots: generative AI's impact on agile role evolution. *J. Softw. Evol. Proc.* **37**, e2740 (2025). <https://doi.org/10.1002/smr.2740>
5. Diebold, P., Ostberg, J., Wagner, S., Zendler, U.: What do practitioners vary in using scrum? In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (eds.) *Agile Processes in Software Engineering and Extreme Programming (XP 2015)*. Lecture Notes in Business Information Processing, vol. 212, pp. 40–51. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18612-2_4
6. Garcia, R., Liao, C.J., Pearce, A., Treude, C.: Gender influence on communication initiated within student teams. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-vol. 1, pp. 432–438 (2022)
7. Nasir, N., Usman, M., Börstler, J., Fogelström, N.D.: Software engineering team project courses with industrial customers: students' insights on challenges and lessons learned. *J. Syst. Softw.* **226**, 112441 (2025)
8. Paasivaara, M., Heikkilä, V.T., Lassenius, C., Toivola, T.: Teaching students Scrum using LEGO blocks. In: Companion Proceedings of the 36th International Conference on Software Engineering (ICSE Companion '14) (2014). <https://doi.org/10.1145/2591062.2591169>
9. Paasivaara, M., Vodă, D., Heikkilä, V.T., Vanhanen, J., Lassenius, C.: How does participating in a capstone project with industrial customers affect student attitudes? In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training, pp. 49–57 (2018)
10. Stabell, I., Kielland Lund, C., Stray, V., Lindsjørn, Y.: Team Composition in Software Engineering Capstone Courses: The Role of Pre-Existing Relationships. In: Proceedings of the 2025 29th International Conference on Evaluation and Assessment in Software Engineering Companion (EASE Companion 2025). Association for Computing Machinery, New York, NY, USA, 235–244 (2025). <https://doi.org/10.1145/3727967.3756827>
11. Stray, V., Moe, N.B., Ganeshan, N., Kobbenes, S.: Generative AI and developer workflows: how GitHub Copilot and ChatGPT influence solo and pair programming. In: Proceedings of the 58th Hawaii International Conference on System Sciences (2025). <https://doi.org/10.24251/HICSS.2025.883>
12. Stray, V., Stabell, I., Sæter, G.E., Barbala, A., Lindsjørn, Y.: Teamwork in agile software development: a mixed-method study of gender diversity and collaboration dynamics. *Inf. Softw. Technol.* **187**, 107840 (2025). <https://doi.org/10.1016/j.infsof.2025.107840>
13. Stray, V., Tkalic, A., Moe, N.B.: The agile coach role: Coaching for agile performance impact. In: Proceedings of the 54th Hawaii International Conference on System Sciences, pp. 6816–6825. AIS Electronic Library (2021). <https://doi.org/10.24251/HICSS.2021.817>
14. Xia, Q., Weng, X., Ouyang, F., Lin, T.J., Chiu, T.K.: A scoping review on how generative artificial intelligence transforms assessment in higher education. *Int. J. Educ. Technol. High. Educ.* **21**(1), 40 (2024)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.





Teaching Agile Methods in K-12 Computer Science Education: An Experience Report on Preparing Teachers to Coach in Classrooms

Konstantin Oltmann^(✉)  and Ralf Romeike 

Freie Universität Berlin, 14195 Berlin, Germany
{konstantin.oltmann,ralf.romeike}@fu-berlin.de

Abstract. Project-based learning with agile methods is gaining ground in school-level computer science (CS) education across primary and secondary schooling (K-12), assigning teachers a new and non-traditional teacher-as-coach role. Despite the growing popularity of this approach, it remains unclear how professional development (PD) should prepare teachers for this role and how teacher learning in such PD can be made explicit.

In response, we report on a theory-informed master’s seminar for pre-service CS teachers that addresses key facets of agility in school contexts. The seminar combined an authentic stakeholder-driven software project with a nested learning arrangement in which participants alternated between experiencing agile development as learners and planning classroom enactment as future teachers. We introduce and apply the agile tetrahedra model as both a design tool and an evaluation lens to connect classroom-level agile content and resources with teacher-relevant outcomes.

Based on instructors’ observations and participants’ written reflections, we distill practical lessons for designing PD that supports teacher-as-coach competencies, strengthens beliefs about the educational value of agile methods, and develops teaching self-efficacy for orchestrating iterative project work under typical school constraints.

Keywords: Agile Education · Project-Based Learning · Teacher Professional Development

1 Introduction

Agile methods are increasingly gaining attention in school-level computer science (CS) education, particularly in secondary education, where they have been discussed and trialed as classroom-adapted practices for student software projects. This emerging relevance is reflected both in computing education research and, in the German context, in formal CS curriculum requirements, as some federal states explicitly include agile project management as mandatory content. Prior

research has proposed and initially evaluated classroom designs and supporting tools for adapting agile principles such as working in iterations with regular feedback and reflection cycles to K-12 project-based learning [10].

Their educational relevance extends beyond domain-specific CS learning goals. Agile project work also addresses competencies associated with volatile, uncertain, complex, and ambiguous (VUCA) environments, where outcomes are difficult to predict and iterative coordination becomes a key capability [2]. This aligns with education policy work, including OECD initiatives [9], which emphasizes learner agency and the ability to navigate complexity and uncertainty as important competencies for future life and work.

This broader relevance also changes what it means to teach agile methods. As agile methods become classroom content, teaching them involves more than procedural routines. It requires teachers to support value-oriented learning and adaptive planning under constraints of time and assessment. All of this calls for a rethinking of the teacher's role, shifting from instructor toward a teacher-as-coach stance [5]. In agile project work, teachers must orchestrate iterative cycles, scaffold productive teamwork and feedback, negotiate scope and quality under tight time constraints, and make both process and product assessable, while still preserving enough student autonomy for meaningful planning and adaptation. These demands make clear that initial teacher education and professional development need explicit support for developing coaching-related competencies, beliefs, and teaching self-efficacy, yet such support remains insufficiently addressed.

In response, this experience report presents a master-level teacher-education seminar in which the proposed agile tetrahedra model served as both a design tool and an evaluation lens, allowing us to make critical learning episodes and teacher-relevant outcomes explicit. After outlining the current state of agile methods in school-level computer science education, we detail the seminar design and implementation. We then synthesize lessons learned from both participants' reflections and the instructors' perspectives, outline future directions, and close with limitations and conclusions.

2 Agile Methods in School CS: What We Know

Research on agile methods in school-level computer science education converges on the view that agile can be productively adapted to classroom project work when practices are selected and reshaped to fit school constraints. Across the literature, agile is typically operationalized through short iterative cycles, visible task management (for example via boards), regular feedback moments, and structured reflection activities such as retrospectives [10].

In published classroom implementations, reported outcomes cluster around students' project and teamwork competencies [4–6]. Studies describe increased student engagement and motivation, improved collaboration and self-management, and gains in problem-solving and computational thinking skills when agile practices are embedded in software projects. At the same time, the

literature indicates that sustainability and fidelity vary by practice. For example, practices that require consistent technical discipline, such as test-first development, can be difficult to maintain in school settings, even when introduced successfully [7]. Overall, the research suggests that teachers routinely adapt agile practices to local conditions rather than implementing a fixed framework unchanged, and that orchestration effort remains a central factor in whether iterative feedback and reflection can be protected within limited lesson time.

This leads directly to a teacher-education gap. While school-based agile project work is commonly described through a teacher-as-coach role, the literature rarely specifies what competencies and professional learning experiences enable teachers to enact this role with confidence, particularly when assessment, pacing, and heterogeneous student prerequisites must be balanced [3]. Existing professional-development work in computer science education remains limited [8], and available models often operate at a single level, which makes it difficult to connect teacher learning, classroom enactment, and contextual constraints in a coherent way. This gap motivates the need for teacher-education designs that make their underlying rationale explicit and provide a structured lens for documenting which learning-triggering episodes occur and what teacher-relevant outcomes emerge.

3 Agile tetrahedra Model

We assume that teachers’ beliefs [11] about the educational value of agile methods and their teaching self-efficacy [13] for coaching agile project work are critical

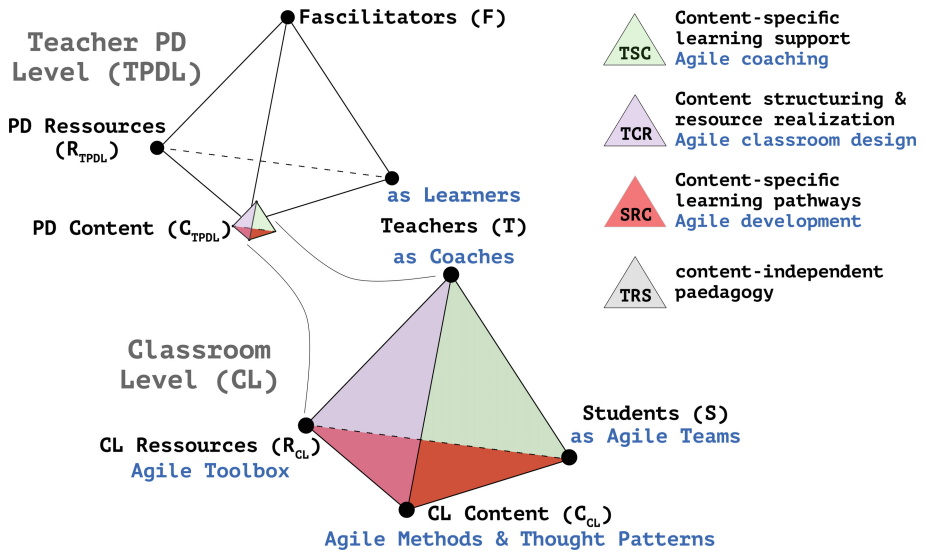


Fig. 1. Proposed Agile Tetrahedra Model

for successful classroom enactment. To support systematic teacher professional development, we therefore introduce a theoretical model that can inform both design and evaluation by linking classroom enactment to teacher-level outcomes.

The model (see Fig. 1) builds on the didactic tetrahedron proposed by Prediger et al. [12], which extends the didactic triangle (teacher, students, content) by adding resources. We define an agile tetrahedron at the classroom level (CL) and add a second level for teacher professional development (TPDL). At CL, agile content (C_{CL}) captures what students should learn about agility (agile thinking, meaningful use of methods, and often coordination around more complex software structures), while agility-supporting resources (R_{CL}) capture what enables this learning (e.g., agile games, method descriptions, project templates, and task boards). Prior work has mainly focused on TCR_{CL} (realizing content through materials and methods) and SCR_{CL} (student collaboration patterns and project outcomes). Our model also foregrounds TSC_{CL} , where the teacher acts as coach and provides content-specific learning support, shaped by beliefs and self-efficacy, represented by the edge TC_{CL} .

The teacher vertex links the two levels. At TPDL, teachers are learners, while at CL they enact coaching. The full CL tetrahedron becomes part of C_{TPDL} , meaning that classroom enactment is itself an object of teacher learning. In our seminar, we operationalized this through a nesting approach in which participants alternated between S_{CL} (acting as learners in an agile team) and T_{CL} (planning teaching moves and scaffolds for school).

4 Seminar Setup

The master-level seminar for pre-service computer science teachers was designed as a 5-ECTS course (approximately 150 h workload) with one weekly 120-minute contact session. It was implemented as a pilot with $n = 3$ participants, comprising two male and one female student at different stages of their individual study trajectories. None of the participants had prior experience with agile project management or agile software development, and two reported comparatively limited programming experience, which required careful pacing and explicit scaffolding. The concept is intended to be established long-term and scaled to larger cohorts.

The seminar pursued four learning objectives, formulated along Bloom's taxonomy [1] to span understanding, application, and transfer. Participants were expected to (1) explain core ideas of agile project management and differentiate key practices across frameworks, (2) apply these practices by conducting an iterative project from a student perspective, (3) use a classroom-realistic programming environment to implement a functional artifact, and (4) analyze and evaluate how agile practices and project work can be transferred into school CS teaching, culminating in the design of concrete enactments including coaching moves and assessment implications.

To connect these objectives, the seminar followed a didactic double-decker¹ rationale. Participants repeatedly engaged in the activities as learners and then shifted to a teacher perspective to discuss how the same elements could be enacted, adapted, and assessed in school. This alternation operationalized the nested design of our Agile tetrahedra model. In particular, it made classroom-level enactment (CL) the immediate object of teacher learning at the professional development level (TPDL), because the same activity was experienced first from the S_{CL} perspective and then reconsidered from the T_{CL} perspective. As a result, participants did not only practice agile routines but also articulated the coaching moves, scaffolds, and assessment decisions required to make these routines instructionally meaningful under typical classroom constraints. In this way, the seminar explicitly supported the role shift toward teacher-as-coach by foregrounding how to scaffold teamwork, feedback, and reflection while preserving student autonomy.

The sequence (see Fig. 2) moved from conceptual grounding to increasingly situated practice. Early weeks focused on guided literature work to establish a shared baseline on agile principles in school-level CS. This was followed by short simulations and agile games, each accompanied by a focused debrief on what was learned and what would be feasible in schools. Among these was a LEGO®-based simulation that makes classical linear project approaches and iterative approaches comparable in a tangible way and, due to its compact duration and structure, is suitable for use in school contexts. Together, these activities addressed aspects such as iterative improvement, requirements clarification, estimation, and retrospectives.

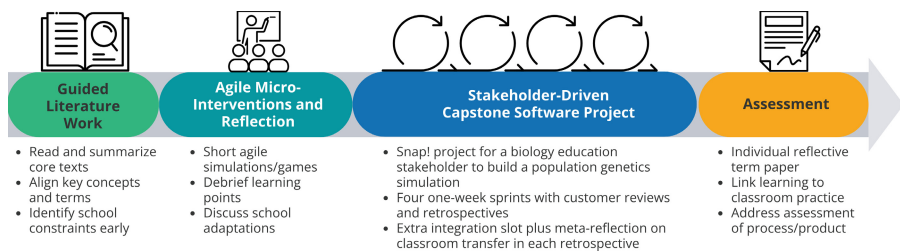


Fig. 2. Seminar structure

The second half centered on a four-week, stakeholder-driven software project aligned with typical school time structures. Implemented in Snap!, the project was commissioned by an external stakeholder from biology education research and aimed to deliver a classroom simulation on population genetics. Using a

¹ Didactic double-decker refers to a seminar design in which pre-service teachers first engage in learning activities from the student perspective and then deliberately switch to a teacher perspective to analyze, adapt, and plan how the same activities could be enacted in school, including appropriate scaffolding and assessment.

fish population as context, the simulation supports exploring changes in allele frequencies and selection effects across generations. Work ran in four one-week sprints, each ending with a customer review and retrospective. To stabilize delivery, the team scheduled an additional integration session shortly before each sprint meeting. Each retrospective included a brief meta-reflection on classroom transfer, focusing on what would need adaptation for teaching and coaching in school. The artifact was enthusiastically accepted by the stakeholder and is intended for future use in biology lessons, providing an authentic outcome beyond the seminar.

Assessment reinforced this transfer focus through an individual term paper in which participants synthesized their learning and connected it to future classroom practice, including the teacher-as-coach role and considerations for assessing both process and product in agile school projects.

5 Lessons Learned and Limitations

Across the pilot, our central assumption was that teachers need to personally experience an agile project before they can credibly coach it in K-12 classrooms. Participants' written reflections strongly supported this view. As one participant wrote, *"the use of the method requires teachers to have experience with agile practices, which can be gained by experiencing such a project from the student perspective"* [translated from German]. This suggests that what mattered was not only knowing agile practices, but having lived through typical tensions of iterative work, such as uncertainty, re-planning, negotiating scope, and aligning team process with product progress. This supports the nesting assumption of the agile tetrahedra model: prospective teachers first need to experience the classroom-level agile setting from the S_{CL} position before they can credibly plan and enact the T_{CL} coaching role.

A key implication is that teacher education must treat school constraints as a first-class design problem. Participants repeatedly questioned feasibility under limited lesson time, which points to "downsizing" as an essential teacher competence, meaning reduced technical scope while protecting feedback and reflection. In our setting, the four-week project succeeded partly because work continued outside the weekly contact session and was stabilized through consistent sprint routines, including a dedicated integration appointment shortly before each sprint meeting. Such time between sessions cannot be assumed in school, so expectation horizons and deliverables must be adjusted accordingly. In model terms, downsizing is not merely a reduction of C_{CL} ; it requires a deliberate reconfiguration of C_{CL} and R_{CL} , so that reduced technical scope is compensated by resources and routines that preserve iteration, feedback, and reflection. A related constraint concerns authentic stakeholder feedback. In many school contexts, it will not be feasible to recruit an external stakeholder for every project, and teachers cannot realistically perform a full stakeholder role repeatedly across multiple teams and iterations. This highlights a practical design challenge for future work in school implementations, namely how to provide credible, timely feedback and

evolving requirements at scale. One promising direction is to explore structured alternatives, including carefully designed proxy roles and, potentially, AI-based systems that can support stakeholder-like feedback while remaining aligned with curricular goals and ethical classroom use. A similar scaling issue will also become relevant if the seminar itself is implemented with larger cohorts, as the small pilot group made intensive feedback and close facilitation comparatively manageable. Future iterations with potentially larger groups will therefore need to test different approaches for organizing feedback, stakeholder interaction, and facilitation at scale.

From the instructors' perspective, the project also revealed a stronger-than-expected technical focus driven by uneven coding experience. This is not merely a pilot artifact but a realistic preview of K-12 conditions, where limited programming fluency often dominates attention and increases coaching demands. The lesson is to prepare teachers for coaching under technical uncertainty, for instance by introducing shared modeling moments and intermediate artifacts that support collective sense-making without reverting to directive instruction.

The pre-project LEGO[®]-based simulation proved essential as a low-threshold primer. It allowed participants to experience core agile dynamics without programming load and made requirement ambiguity and communication gaps tangible, thereby creating shared reference points for later project episodes. Given the heterogeneity of school contexts, any concrete setup will require local adaptation, particularly with regard to class size and grade level. Nevertheless, an early, experience-based simulation of this kind should remain a stable component.

Overall, the seminar design functioned well and appears suitable for institutionalization in future semesters. The ideas underlying the seminar can also inform shorter in-service teacher trainings, although the limited timeframe requires a substantial reduction and prioritization of activities. At the same time, this pilot was based on a very small cohort and a single project context, which limits the breadth of experiences and the generalizability of lessons. These constraints motivate iterative refinement and replication across upcoming semesters with larger cohorts and varied project settings, while preserving the seminar's core strength: combining lived student-perspective experience with repeated, transfer-oriented reflection on teaching and coaching agile project work in school contexts.

6 Conclusion and Future Work

In this experience report, we described a master-level seminar that prepares pre-service computer science teachers to teach agile methods in school contexts by systematically positioning them in a teacher-as-coach role. Our main takeaway is that this role cannot be developed through conceptual instruction alone. Instead, teachers benefit from personally experiencing an agile project, including the uncertainties, coordination demands, and reflection cycles that make agility meaningful in practice. This lived experience provides a credible foundation for later didactic transformation, because it equips future teachers to

scaffold teamwork, protect feedback and reflection under classroom constraints, and make process-oriented learning visible. Building on the encouraging results of this pilot, we will refine and scale the seminar in future semesters and complement experience-based evaluation with an empirical pre-post design targeting teacher beliefs and teaching self-efficacy.

Acknowledgments. We used ChatGPT 5.5 and DeepL Write to support language refinement in this experience report. We take full responsibility for the final form of the manuscript.

Teaching Material. The teaching materials and resources used are available from the first author upon request.

References

1. Anderson, L.W., Krathwohl, D.R.: A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Allyn & Bacon, New York (2001)
2. Bennett, N., Lemoine, G.J.: What VUCA really means for you. *Harvard Business Review* **92**(1/2) (2014)
3. Fitsilis, P., Boti, E., Damasiotis, V.: From Agile Principles to Agile Classrooms: A Structured Curriculum for Educator Upskilling. *Int. J. Changes Educ.* (2025)
4. Fronza, I., Pahl, C., Susanj, B.: Agile Methods Make It to Non-vocational High Schools. In: *Computer Supported Education*. vol. 1220, pp. 355–372. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-58459-7_17
5. Kastl, P., Romeike, R.: Agile projects to foster cooperative learning in heterogeneous classes. In: *2018 IEEE Global English Education Conference (EDUCON)*, pp. 1182–1191 (2018)
6. Missiroli, M., Russo, D., Ciancarini, P.: Agile for Millennials: a Comparative Study. In: *2017 IEEE/ACM 1st International Workshop on SW Eng. Curricula for Millennials*, pp. 47–53. IEEE, Buenos Aires, Argentina (2017)
7. Missiroli, M., Russo, D., Ciancarini, P.: Teaching Test-First Programming: Assessment and Solutions. In: *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, pp. 420–425. IEEE, Turin (2017). <https://doi.org/10.1109/COMPSAC.2017.229>
8. Mouza, C., Coddling, D., Pollock, L.: Investigating the impact of research-based professional development on teacher learning and classroom practice: Findings from computer science education. *Comput. Educ.* **186**, 104530 (2022)
9. OECD: Embedding Values and Attitudes in Curriculum: Shaping a Better Future. OECD (2021). <https://doi.org/10.1787/aee2adcd-en>
10. Oltmann, K., Romeike, R.: Doing versus being agile in k-12 computer science education: Moving from methodology to mindset. In: *Proceedings of IFIP TC3 Open Conference on Computers in Education, OCCE 2025, Rabat, Morocco (in press)* (2026)
11. Pajares, M.F.: Teachers' beliefs and educational research: cleaning up a messy construct. *Rev. Educ. Res.* **62**(3), 307–332 (1992)

12. Prediger, S., Roesken-Winter, B., Leuders, T.: Which research can support PD facilitators? Strategies for content-related PD research in the three-tetrahedron model. *J. Math. Teacher Educ.* **22**(4), 407–425 (2019)
13. Tschannen-Moran, M., Hoy, A.W., Hoy, W.K.: Teacher efficacy: its meaning and measure. *Rev. Educ. Res.* **68**(2), 202–248 (1998)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Posters Track



Making Technical Debt Visible: An Indicator-Driven Radar for Agile Teams

Mateus Costa Lucena¹(✉) , Antonio Carlos Pacagnella Junior² ,
and Gizelle Sandrini Lemos¹

¹ Instituto de Pesquisas Eldorado, Campinas, Brazil
{mateus.lucena,gizelle.lemos}@eldorado.org.br

² Universidade de São Paulo, São Paulo, Brazil
acpjr@usp.br

Abstract. Agile teams frequently struggle to monitor technical quality, especially in innovation-driven environments where decisions are made under uncertainty and time pressure. Although several metrics exist, most organizations lack a structured approach for defining, tracking, and acting on quality indicators. This work proposes a visual quality management process based on the segmentation of the Technical Debt Radar, enabling teams to map indicators to relevant dimensions and integrate them into Scrum practices. The model supports continuous measurement, enhances communication, and explicitly connects indicators to Scrum decision-making, enabling technical debt to be discussed and managed rather than merely perceived.

Keywords: Technical Debt · Scrum · Visual Quality Management

1 Introduction

Ensuring technical quality in agile software projects remains a challenge, particularly in innovation-driven environments where requirements evolve rapidly and architectural decisions carry long-term consequences. Although quality metrics are widely discussed, teams often struggle to define indicators that are measurable, actionable, and aligned with project objectives. As a result, decisions on refactoring or technical debt mitigation frequently rely on subjective judgment rather than systematic evidence [1].

The Technical Debt Radar proposed by Caroli and Pinto [2] offers a visual tool for discussing technical commitments, but its use is traditionally limited to qualitative conversations. Building on this foundation, this work proposes a structured segmentation of the radar based on quality dimensions inspired by the findings reported by Tonin [3]. By mapping indicators to functional and delivery-oriented dimensions, the model transforms the radar into a visual quality management process that can be integrated into Scrum artifacts and events.

Although widely used in agile practice, the original Technical Debt Radar has not been formally extended with measurable indicators or aligned with frameworks that support empirical process control, such as Scrum. This work addresses

that gap by proposing an operational model that combines measurable indicators with a visual governance structure embedded in Scrum events.

The proposed approach contributes theoretically by framing visual quality control as an extension of Scrum’s empirical pillars—transparency, inspection, and adaptation [4]. Practically, it provides teams with a reproducible method for defining indicators, monitoring trends, and prioritizing technical debt.

This work is guided by the following research question: “*How can a structured set of technical quality indicators be integrated into Scrum events and artifacts to support product health monitoring and technical debt management without compromising agility?*”

The remainder of this paper is structured as follows: Sect. 2 presents the Visual Quality Management Framework, detailing the indicator model and the segmented radar structure. Section 3 discusses conclusions and future work, including directions for empirical validation and practitioner evaluation.

2 Visual Quality Management Framework

2.1 Technical Quality Indicator Model

Technical quality indicators should be defined at the beginning of the project, ideally during “Sprint 0” or while establishing the Definition of Done (DoD). At this stage, teams reflect on the essential question: “*What does technical quality mean for this project?*”

Based on this discussion, an initial set of indicators is selected. To ensure clarity and consistency, each indicator follows a standard documentation structure that specifies what is measured, how data is collected, target values, and data sources. This structure guarantees actionability and alignment with project goals. However, these indicators are not static; they should evolve throughout the project lifecycle as architecture matures, risks emerge, and business priorities shift.

Figure 1 presents an example of how a team might document a web performance indicator.

Indicator	Load Time (LCP – Largest Contentful Paint)
Description	Measures the time required for the largest content element on the main page to be rendered. Crucial for user experience and SEO.
Formula / Collection Method	Time in seconds measured from the start of navigation until the largest visible element in the viewport is rendered.
Target / Threshold	LCP < 2,5 seconds.
Measurement Tool	Google PageSpeed Insights / Lighthouse / Sentry.
Measurement Frequency	At every production deployment and monitored continuously.

Fig. 1. Illustrative example of the documentation format applied to a web indicator.

By adopting this model, teams create a living indicator system that evolves as the product does, allowing continuous refinement of thresholds, metrics, and monitoring scope.

2.2 Adaptive Technical Debt Radar

Beyond indicator definition and Scrum integration, this work introduces a segmented radar structure that enables visual governance of technical quality.

These indicators are projected onto a segmented radar, allowing teams to visualize quality trade-offs and debt accumulation patterns over time.

Figure 2 illustrates the model.

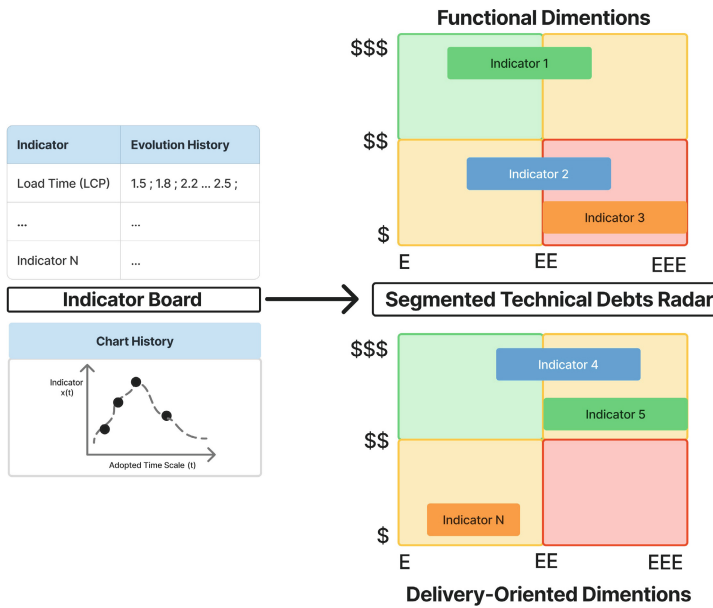


Fig. 2. Segmented Technical Debt Radar.

Functional dimensions (performance, security, scalability, availability, and reliability) represent system behavior.

Delivery-oriented dimensions (maintainability, evolvability, testability, deployability) reflect development flow and architectural maturity.

By combining both perspectives, the radar enables identification of critical areas and ongoing quality monitoring across iterations. For example, a decline in maintainability (delivery-oriented dimension) may increase the likelihood of performance degradation (functional dimension), explicitly linking internal quality erosion to externally observable technical debt. This interdependence helps teams reason about trade-offs and anticipate long-term consequences of short-term decisions.

2.3 Integration Into Scrum Framework

Once defined, indicators are incorporated into Scrum through its empirical pillars: transparency, inspection, and adaptation.

Transparency is achieved by referencing indicators in acceptance criteria. Even if compliance is postponed and treated as accepted technical debt, the indicator is still tracked.

As illustrated in Fig. 3, the visual mapping process explicitly connects indicators to Scrum artifacts, reinforcing transparency and enabling traceability between quality metrics and backlog decisions. The framework embeds quality discussions into routine inspection and adaptation events, making deviations explicit and enabling early detection of quality drift.

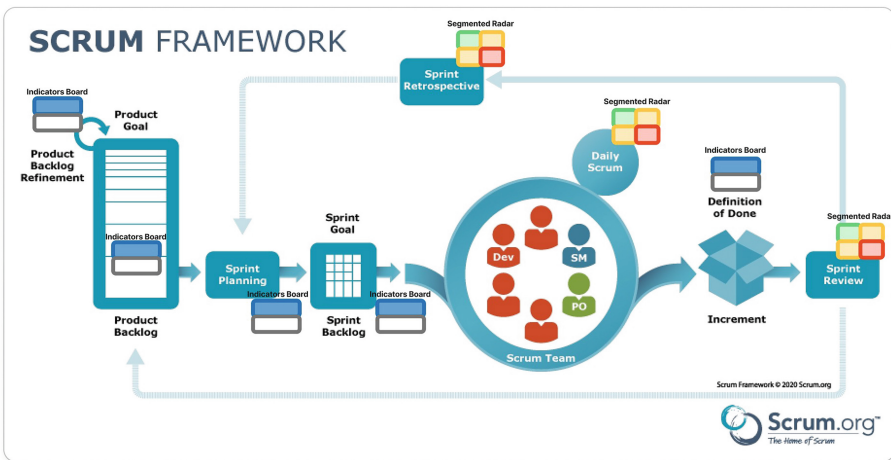


Fig. 3. Indicator mapping process in Scrum (adapted from scrum.org poster [5]).

Inspection occurs within regular events: alerts may appear in stand-ups; trends are discussed during reviews; and root causes are examined in retrospectives.

Adaptation takes place when backlog priorities, acceptance criteria, or refactoring actions change based on indicator behavior, facilitating data-driven decisions during sprint reviews and retrospectives.

3 Conclusions and Future Work

The absence of formal quality indicators often transforms technical debt into subjective debate rather than objective management. The segmented Technical Debt Radar integrates indicators into Scrum and enables a continuous cycle of measurement, inspection, and adaptation. The approach improves communication, supports prioritization, and aligns innovation with reliability.

Future work includes empirical validation in industrial contexts and the integration of human-centric indicators to support reflection on team performance.

References

1. Irgens, M., Martini, A.: Visualization usage in technical debt management. In: Peter, S., Kropp, M., Aguiar, A., Anslow, C., Lunesu, M.I., Pinna, A. (eds.) XP 2025. LNBIP, vol. 545, pp. 186–194. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-94544-1_13
2. Pinto, A., Caroli, P.: Lean Delivery com Métricas Ágeis. Caroli.org (2022)
3. Tonin, G.S.: Technical Debt Management in the Context of Agile Methods in Software Development. PhD thesis, University of São Paulo (2018)
4. Schwaber, K., Sutherland, J.: The Scrum Guide (2020). <https://scrumguides.org/scrum-guide.html>. Accessed 10 May 2026
5. Scrum.org: Scrum Framework Poster. <https://www.scrum.org/resources/scrum-framework-poster>. Accessed 22 Mar 2026

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Author Index

A

Almås, Steffen 171

B

Bakke, Iselin Mordal 111

Berntzen, Marthe 61, 171

Broomandi, Fateme 123

D

da Silva, Tiago Silva 3

Diebold, Philipp 31

E

Eilers, Karen 3

El Outmani, Moustapha 80

F

Fagerstrøm, Asle 143

G

Geger, Tobias 13, 155

Goknil, Arda 100

Grøneng, Camilla 61

Gundelsby, Jan Henrik 61, 91

H

Hatahet, Ahmad 80

Hinderks, Andreas 71

Hope, Maria 133

I

Inkermann, David 155

J

Junior, Antonio Carlos Pacagnella 193

K

King, Benjamin 80

Kniep, Tim Niklas 80

L

Lemos, Gizelle Sandrini 193

Lindsjörn, Yngve 111, 171

Lucena, Mateus Costa 193

Lund, Camilla Kielland 171

M

Martini, Antonio 91

Mo, Elisabeth 143

Moe, Nils Brede 51, 133

Molléri, Jefferson Seide 143

N

Neumann, Michael 3, 162

Nguyen, Phu 91

Nodari, Alexandre 22

Nunes, Leonardo 22

O

Ochoa, Lina 91

Olafsen, Henrik Brunvatne 100

Oltmann, Konstantin 181

Orucevic, Mili 91

P

Paasivaara, Maria 123

Pellenz, Marcelo E. 22

R

Raddatz, Thomas 80

Rausch, Andreas 13, 80, 155

Rauschenberger, Maria 3

Romeike, Ralf 181

Rook, Anna 40

S

Scalabrin, Edson Emilio 22
Schiering, Ina 13
Schön, Eva-Maria 3, 71, 162
Schreiter, Bente 40
Sen, Sagar 100
Senapathi, Mali 3
Shenoy, Manthan Venkataramana 80
Silva da Silva, Tiago 162
Stabell, Idunn 171
Stenzel, Frauke 13
Stray, Viktoria 51, 111, 133, 171

T

Thomaschewski, Jörg 71

V

Verwijs, Christiaan 40
Videsjorden, Adela Nedisan 100
Vigeland, Dan 61

W

Walter, Marcelo Luis 22
Willumsen, Mathias 133
Wittek, Stefan 13, 155